# 68' MICRO JOURNAL

Australia A $7.95
Singapore S $14.95
Malaysia M $14.25
New Zealand NZ $9.85
Hong Kong H $29.00
Sweden 30:-SEK

$2.95USA

## OS-9 Atari Amiga Mac S-50

**6800 6809 68008 68000 68010 68020 68030**

The Magazine for Motorola CPU Devices *For Over a Decade!*

This Issue:

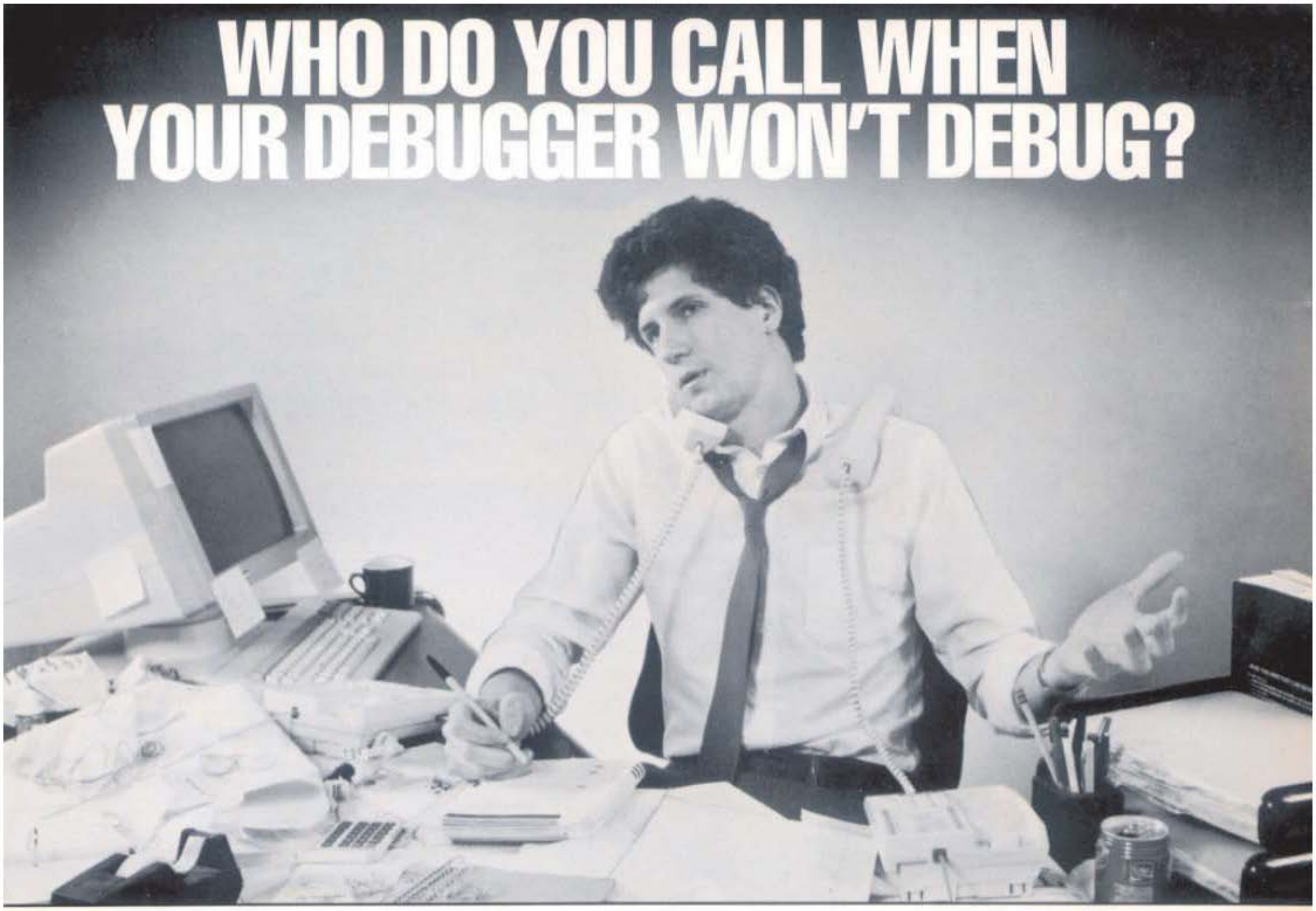
OS-9 SK*DOS Atari Amiga FLEX Macintosh *A User Contributor Journal* **And Lots More!**

VOLUME XI ISSUE III ● Devoted to the 68XXX User ● March 1989

The Grandfather of "DeskTop Publishing™"

0 74470 12810 03

000422 A/E MJ
MR. MICKEY FERGUSON
P. O. BOX 87
KINGSTON SPRINGS TN 37082



# WHO DO YOU CALL WHEN YOUR DEBUGGER WON'T DEBUG?

The problem with most real-time operating systems is simple, they're not an integrated solution. You end up dealing with a multitude of suppliers for languages, compilers, debuggers and other important development tools. And when something does go wrong, it can be a frustrating experience trying to straighten out the mess.

### *Why Not Try the Microware One-Stop Total Solution?*

Microware's OS-9 Real-Time Operating System is a total integrated software system, not just a kernel. We offer an extensive set of development tools, languages, I/O and Kernel options. ***And this total integrated solution is entirely designed, built and supported by the same expert Microware team.***

Microware is a registered trademark of Microware Systems Corporation.
OS-9 is a trademark of Microware.
UNIX is a trademark of AT&T.
VAX is a trademark of DEC.

### The OS-9 Success Kit

*A Total Integrated Solution for Your Next Project*

**Development Tools:**

C Source Level Debugger
Symbolic Debugger
System State Debugger
uMACS Text Editor
Electronic Mail
Communications
Super Shell

**Kernel Options:**

MMU (Security Protection) Support
Math Coprocessor Support

**Languages:**

C*
Basic
Pascal
Fortran
Ada**
Assembler*

**I/O Options:**

SCSI, SASI & SMD Disks
3-, 5-, 8-inch Diskettes
Magnetic Tape
Ethernet - TCP/IP
Arcnet - OS-9/Net

*Resident or UNIX versions available
**VAX hosted

### *Modularity Lets YOU Choose Just What You Need.*

The modular design of OS-9 allows our Operating System to adapt as your requirements change. OS-9 can support a complete spectrum of applications — from embedded ROM-based code in board-level products all the way up to large-scale systems.

### *Support is Part of the Package.*

Microware is proudly setting the industry's standard for customer support. You'll find professional and comprehensive technical documentation and a Customer Hotline staffed by courteous and authoritative software engineers.

So stop messing with simple kernels and independent suppliers. Call Microware today and find out more about the "One-Stop Integrated Solution" with OS-9!

microware® OS-9

**Microware Systems Corporation**
1900 N.W. 114th Street
Des Moines, Iowa 50322
Phone: 515/224-1929

**Western Regional Office**
4401 Great America Parkway
Santa Clara, California 95054
Phone: 408/980-0201

**Microware Japan Ltd.**
41-19 Honcho 4-Chome
Funabashi City
Chiba 273, Japan
Phone: 0474 (22) 1747

## Mustang-020 Mustang-08 Benchmarks

| | 32 bit Integer | Register Long |
|---|---|---|
| IBM AT 7300 Xenix Sys 3 | 9.7 | |
| AT&T 7300 UNIX PC 68010 | 7.2 | 4.3 |
| DEC VAX 11/780 UNIX Berkley 4.2 | 3.6 | 3.2 |
| DEC VAX 11/750 " " | 5.1 | 3.2 |
| 68008 OS-9 68K 8 Mhz | 18.0 | 9.0 |
| 68000 OS-9 68K 10 Mhz | 6.5 | 4.0 |
| *MUSTANG-08 68008 OS-9 68K 10 Mhz* | *9.8* | *6.3* |
| *MUSTANG-020 68020 OS-9 68K 16 Mhz* | *2.2* | *0.88* |
| *MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz* | *1.8* | *1.22* |

```
Main()
            {
                register long i;
                for (i=0; i < 999999; ++i);
            }
```

**Estimated MIPS - MUSTANG-020 ..... 4.5 MIPS,**

**Burst to 8 - 10 MIPS: Motorola Specs**

## OS-9

| | |
|---|---|
| OS-9 Professional Ver | $850.00 |
| *Includes C Compiler | |
| Basic09 | 300.00 |
| C Compiler | 500.00 |
| 68000 Disassembler (w/source add: $100.00) | 100.00 |
| Fortran 77 | 750.00 |
| Microware Pascal | 500.00 |
| Omegasoft Pascal | 900.00 |
| Stylo-Graph | 495.00 |
| Stylo-Spell | 195.00 |
| Stylo-Merge | 175.00 |
| Stylo-Graph-Spell-Merge | 695.00 |
| PAT w/C source | 229.00 |
| JUST w/C source | 79.95 |
| PAT/JUST Combo | 249.50 |
| Sculptor+ (see below) | 995.00 |
| COM | 125.00 |

## UniFLEX

| | |
|---|---|
| UniFLEX (68020 ver) | $450.00 |
| Screen Editor | 150.00 |
| Sort-Merge | 200.00 |
| BASIC/PreCompiler | 300.00 |
| C Compiler | 350.00 |
| COBOL | 750.00 |
| CMODEM w/source | 100.00 |
| TMODEM w/source | 100.00 |
| X-TALK (see Ad) | 99.95 |
| Cross Assembler | 50.00 |
| Fortran 77 | 450.00 |
| Sculptor+ (see below) | 995.00 |

| | |
|---|---|
| Standard MUSTANG-020™ shipped 12.5 Mhz. | |
| Add for 16.6 Mhz 68020 | 75.00 |
| Add for 16.6 Mhz 68881 | 375.00 |
| Add for 20 Mhz 68020/RAM | 750.00 |
| 16 Port exp. RS-232 | 335.00 |
| Requires 1 or 2 Adapter Cards below RS232 Adapter | 165.00 |
| Each card supports 4 additional ser. ports (total of 36 serial ports supported) | |
| 60 line Parallel I/O card | 398.00 |
| Uses 3 68230 Interface/Timer chips, 6 groups of 8 lines each, separate buffer direction control for each group. | |
| Prototype Board | 75.00 |
| areas for both dip and PGA devices & a pre-wired memory area up to 512K DRAM. | |
| SBC-AN | 475.00 |
| Interface between the system and ARCNET modified token-passing LAN, fiber optics optional - call. | |
| LAN software drivers | 120.00 |
| Expansion for Motorola I/O Channel Modules | $195.00 |
| Special for complete MUSTANG-020™ system buyers - Sculptor+ $695.00. SAVE $300.00 | |
| Software Discounts | |

All MUSTANG-020™ system and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quotes. Discounts apply after the sale as well.

Note: Only Professional OS-9 Now Available (68020 Version) Includes ($500) C Compiler - 68020 & 68881 Supported - For UPGRADES Write or Call for Professional OS-9 Upgrade Kit

# The P R O !

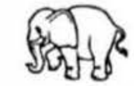
## Mustang Specifications

12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
32-bit wide data and address buses, non-multiplexed
on chip instruction cache
object code compatible with all 68XXX family processors
enhanced instruction set - math co-processor interface
68881 math hi-speed floating point co-processor (optional)
direct extension of full 68020 instruction set
full support IEEE P754, draft 10.0
transcendental and other scientific math functions
2 Megabyte of SIP RAM (512 x 32 bit organization)
up to 256K bytes of EPROM (64 x 32 bits)
4 Asynchronous serial I/O ports standard
optional to 20 serial ports
standard RS-232 interface
optional network interface
buffered 8 bit parallel port (1/2 MC68230)
Centronics type pinout
expansion connector for I/O devices
16 bit data path
256 byte address space
2 interrupt inputs
clock and control signals
Motorola I/O Channel Modules
time of day clock/calender w/battery backup
controller for 2, 5 1/4" floppy disk drives
single or double side, single or double density
35 to 80 track selectable (48-96 TPI)
SASI interface
programmable periodic interrupt generator
interrupt rate from micro-seconds to seconds
highly accurate time base (5 PPM)
5 bit sense switch, readable by the CPU
Hardware single-step capability

*Don't be mislead! ONLY Data-Comp delivers the Super MUSTANG-020*

These hi-speed 68020 systems are presently working at NASA, Atomic Energy Commission, Government Agencies as well as Universities, Business, Labs, and other Critical Applications Centers, worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level V compatability and low cost is a must.

Only the "PRO" Version of OS-9 Supported!



This is *HEAVY DUTY* Country!

For a limited time we will offer a $400 trade-in on your old 68XXX SBC. Must be working properly and complete with all software, cables and documentation. Call for more information

**Price List:**

| | | |
|---|---|---|
| Mustang-020 SBC | | $2490.00 |
| Cabinet w/switching PS | | $299.95 |
| 5"-80 track floppy DS/DD | | $269.95 |
| Floppy Cable | | $39.95 |
| OS-9 68K Professional Version | | $850.00 |
| C Compiler ($500 Value) | | N/C |
| Winchester Cable | | $39.95 |
| Winchester Drive 25 Mbyte | | $895.00 |
| Hard Disk Controller | | $395.00 |
| Shipping USA UPS | | $20.00 |
| UniFLEX | Less | $100.00 |
| MC68881 f/p math processor | Add | $275.00 |
| 16.67 Mhz MC68020 | | $375.00 |
| 16.67 Mhz MC68881 | | $375.00 |
| 20 Mhz MC68020 Sys | | $750.00 |
| Note all 68881 chips work with 20 Mhz Sys | | |
| | Total: | $5299.80 |

# NEW LOWER PRICES

## 25 Mbyte HD ~~$4299.80~~ $3749.80
## 85 Mbyte HD ~~$5748.80~~ $4548.80

**Data-Comp Division**
A Decade of Quality Service"
*Systems World-Wide*
**Computer Publishing, Inc.** 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

**A Member of the CPI Family**

# 68 Micro Journal

*10 Years of Dedication to Motorola CPU Users*

6800 6809 68000 68010 68020

The Originator of "DeskTop Publishing™"

**Publisher**
Don Williams Sr.

**Executive Editor**
Larry Williams

**Production Manager**
Tom Williams

**Office Manager**
Joyce Williams

**Subscriptions**
Cheryl Hodge

***Contributing & Associate Editors***

Ron Anderson
Ron Voigts
Doug Lurie
Ed Law
Dr. E.M. "Bud" Pass
Art Weller
Dr. Theo Elbert
& Hundreds More of Us

## Contents

68 MICRO JOURNAL

*"Contribute Nothing - Expect Nothing"* DMW 1986

***COMPUTER PUBLISHING, INC.***

*"Over a Decade of Service"*

"World Wide"

**68 MICRO JOURNAL**
**Computer Publishing Center**
**5900 Cassandra Smith Road**
**P● Box 849**
**Hixson, TN 37343**
**Phone (615) 842-4600 Telex 510 600-6630**

Copyrighted © 1987 by Computer Publishing, Inc.

68 Micro Journal is the *original* "DeskTop Publishing" product and has continously published since 1978 using only micro-computers and special "DeskTop" software. Using first a kit built 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 Micro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

68 Micro Journal (ISSN 0194-5025) is published 12 times a year by Computer Publishing Inc. Second Class Postage paid at Hixson, TN. and additional entries. POSTMASTER: send address changes to 68 Micro Journal, POB 849, Hixson, TN 37343.

### Subscription Rates

1 Year $24.50 USA, Canada & Mexico $34.00 a year. Others add $12.00 a year surface, $48.00 a year Airmail, USA funds. 2 years $42.50, 3 years $64.50 plus additional postage for each additional year.

### Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles submitted should be on diskette, OS-9, SK*DOS, FLEX, Mac`ntosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

*Please - do not format with spaces any text indents, charts, etc. (source listing o.k.). We wi l edit in all formatting. Text should fa  flush left and use a carriage return only to indicate a paragraph end. Please write for fr ee authors guide.*

### Letters & Advertising Copy

Letters to the Editor should be the original copy, signed! Letters of grip as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of $15.50 for first 15 words. Add $.60 per word thereafter. No classifieds accepted by telephone.

# PAT - JUST

## PAT

With 'C' Source

$229.00

All OS-9
68XXX
Systems

service

For Over a Decade!
1,200 Titles +

A Division of
CPI

S.E.
Media

5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

6800-6809-68008-68000-68010-68020

PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved PIE, this is for YOU! All PIE features & much more! Too many features to list. And if you don't like ours, change or add your own. C source included. Easily configured to your CRT terminal, with special configuration section. No sweat!

**68008 - 68000 - 68010 - 68020 OS-9 68K $229.00**

# COMBO

# PAT/JUST

# Special $249.00

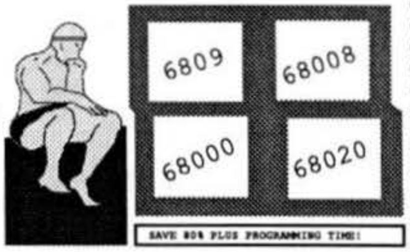
## JUST

**JUST from S. E. MEDIA - -** Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

**68008 - 68000 - 68010 - 68020 OS-9 68K**
**With 'C' source $79.95**

# CLOSE OUT SPECIAL

# SCULPTOR



**From the world's oldest**
**& largest OS-9 software house!**

## CUTS PROGRAMMING TIME UP TO 80%

## 6809/68000-68030 Save 90%

SCULPTOR-a 4GL - Only from S.E. Media at these prices. OS-9 levels one and two (three GIMIX) 6809, all 68XXX OS-9 standard systems. Regular SCULPTOR versions 1.4:6. One of if not the most efficient and easy to develope DBMS type systems running under OS-9. A system of flexible keyed file access that allows extremely fast record and data retrieval, insertion and deletion or other programmed modifications. Access by key or in ascending order, very fast. The system provides automatic menu generation, compilation and report generation. Practically unlimited custom input format and report formatting. A rich set of maintenance and repair utilities. An extremely efficient development environment that cuts most programming approximately 80% in development and debugging! Portable, at source level, to MS-DOS, UNIX and many other languages and systems.

Standard Version: 1.14:6 6809 - $1295.00
68000 $1295.00
68020 $1990.00

Due to a "Special One Time" Purchase, We Are Making This Savings Offer. Quantities Limited!

*Once this supply is gone the price goes back up!*

System OS-9: 6809/68000-68030

• Regular $~~1295.00~~

ONLY $99.95

**S.E. MEDIA**
POB 849 5900 CASSANDRA SMITH ROAD
HIXSON, TN 37343 615 842-4601
TELEX 510 600-6630

$AVE - WHILE SUPPLIES LAST!

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

SOFTWARE

!!! Please Specify Your Operating System and Disk Size !!!

# SCULPTOR

**Full OEM & Dealer Discounts Available!**

### THE SCULPTOR SYSTEM

Sculptor combines a powerful fourth generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

### AN ESTABLISHED LEADER

Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever-increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

### SYSTEM INDEPENDENCE

Sculptor is available on many different machines and for most operating systems, including MS DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user micros up to large minis and mainframes. Sculptor is constantly being ported to new systems.

### APPLICATION PORTABILITY

Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and – without any alterations to the programs – run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high-speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

### SPEED AND EFFICIENCY

Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

### INTERNATIONALLY ACCEPTED

By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australasia, the Americas and Europe – Sculptor is already at work in over 20 countries.

### THE PACKAGE

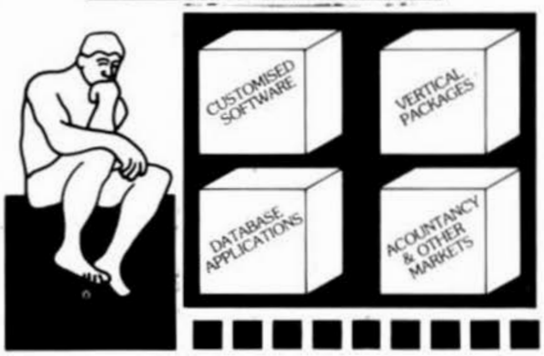
With every development system you receive:

- □ A manual that makes sense
- □ A periodic newsletter
- □ Screen form language
- □ Report generator
- □ Menu system
- □ Query facility
- □ Set of utility programs
- □ Sample programs

For resale products, the run-time system is available at a nominal cost.

**Facts** **Features**

**Sculptor for 68020**
**OS-9 & UniFLEX**
**$995**



### DATA DICTIONARY

Each file may have one or more record types described. Fields may have a name, heading, type, size, format and validation list. Field type may be chosen from:

- □ alphanumeric
- □ integer
- □ floating point
- □ money
- □ date

### DATA FILE STRUCTURE

- □ Packed, fixed-length records
- □ Money stored in lower currency unit
- □ Dates stored as integer day numbers

### INDEXING TECHNIQUE

Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

### INPUT DATA VALIDATION

Input data may be validated at three levels:

- □ automatic by field type
- □ validation list in data dictionary
- □ programmer coded logic

### ARITHMETIC OPERATORS

| | |
|---|---|
| – | Unary minus |
| * | Multiplication |
| / | Division |
| % | Remainder |
| + | Addition |
| – | Subtraction |

### RELATIONAL OPERATORS

| | |
|---|---|
| = | Equal to |
| < | Less than |
| > | Greater than |
| < = | Less than or equal to |
| > = | Greater than or equal to |
| < > | Not equal to |
| and | Logical and |
| or | Logical or |
| ct | Contains |
| bw | Begins with |

### SPECIAL FEATURES

- □ Full date arithmetic
- □ Echo suppression for passwords
- □ Terminal and printer independence
- □ Parameter passing to sub-programs
- □ User definable date format

### MAXIMA AND MINIMA

| | |
|---|---|
| Minimum key length | 1 byte |
| Maximum key length | 160 bytes |
| Minimum record length | 3 bytes |
| Maximum record length | 32767 bytes |
| Maximum fields per record | 32767 |
| Maximum records per file | 16 million |
| Maximum files per program | 16 |
| Maximum open files | Operating system limit |

### PROGRAMS

- □ Define record layout
- □ Create new indexed file
- □ Generate standard screen-form program
- □ Generate standard report program
- □ Compile screen-form program
- □ Compile report program
- □ Screen-form program interpreter
- □ Report program interpreter
- □ Menu interpreter

- □ Query facility
- □ Reformat file
- □ Check file integrity
- □ Rebuild index
- □ Alter language and date format
- □ Setup terminal characteristics
- □ Setup printer characteristics

### SCREEN-FORM LANGUAGE

- □ Programmer defined options and logic
- □ Multiple files open in one program
- □ Default or programmable processing of exception conditions
- □ Powerful verbs for input, display and file access
- □ Simultaneous display of multiple records
- □ Facility to call sub-programs and operating system commands
- □ Conditional statements
- □ Subroutines
- □ Independence of terminal type

**MUSTANG-020 Users - Ask For Your Special Discount!**

| | * | ** | *** | | * | ** | *** |
|---|---|---|---|---|---|---|---|
| **MUSTANG-020** | *$1,990 | $398 | $795 | **PC/XT/AT/MSDOS** | $695 | $139 | $299 |
| **MUSTANG-08** | *$1,295 | $259 | $495 | **Call or write for prices on the following systems.** | | | |

XENIX SYS III & V, MS-NET, UNIX SYS III & V, ATARI OS-9, 68K, UNOS, ULTRIX/VMS (VAX.REGAL), STRIDE, ALTOS, APRICORT, ARETE, ARMSTRONG, BLEASDALE, CHARLES RIVERS, GMX, CONVERG.TECH, DEC, CIFER, EQUINOX, GOULD, HP, HONEYWELL, IBM, INTEL, MEGADATA, MOTOROLA, NCR, NIXDORF, N.STAR, OLIVETTI/AT&T, ICL, PERKINS ELMER, PHILLIPS, PIXEL, PLESSEY, PLEXUS, POSITRON, PRIME, SEQUENT, SIEMENS, SWTPC, SYSTIME, TANDY, TORCH, UNISYS, ZYLOG, ETC.

*** For SPECIAL LOW SCULPTOR prices especially for 6809/68XXX OS-9 Systems - See Special Ad this issue. Remember, "When they are gone the price goes back up as above!"**

**... Sculptor Will Run On Over 100 Other Types of Machines ...**

**... Call for Pricing ...**

**!!! Please Specify Your Make of Computer and Operating System !!!**

* Full Developement Package
** Run Time Only
*** C Key File Library

**Availability Legends**
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC9 = Color Computer OS-9
CCF = Color Computer FLEX

VISA

**South East Media**
5900 Cassandra Smith Rd. - Hixson, Tn. 37343
Telephone: (615) 842-4600 Telex: 5106006630

master card

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

C

*The C Programmers Reference Source. Always Right On Target!*

# C User Notes

## A Tutorial Series

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
*Computer Systems Consultants*

### INTRODUCTION

This chapter begins with a statement by Nicholas Wirth about his new language, Oberon. It then discusses the problems of linking C, Fortran, and Pascal functions together in order to form an executable program.

### FROM MODULA TO OBERON

The following discussion was taken almost verbatim from a statement by Nicholas Wirth concerning his new programming language, which he calls Oberon. For those unfamiliar with Wirth, he is credited with the initial design of Pascal and of its successors Modula and Modula-2.

It is included here because it illustrates the fact that successors of the Pascal language are becoming much more like the C language than are later revisions of the C language becoming more like Pascal or Modula.

The programming language Oberon is the result of a concentrated effort to increase the power of Modula-2 and simultaneously to reduce its complexity. Several features were eliminated, and a few were added in order to increase the expressive power and flexibility of the language. This paper describes and motivates the changes. The language is defined in a concise report.

The programming language Oberon evolved from a project whose goal was the design of a modern, flexible and efficient operating system for a single-user workstation. A principal guideline was to concentrate on properties that are genuinely essential and, as a consequence, to omit ephemeral issues. It is the best way to keep a system in hand, to make it understandable, explicable, reliable and efficiently implementable.

Initially, it was planned to express the system in Modula-2 [1], as that language supports the notion of modular design quite effectively, and because an operating system has to be designed in terms of separately compilable parts with conscientiously chosen interfaces. In fact, an operating system should be no more than a set of basic modules, and the design of an application must be considered as a goal-oriented extension of that basic set. Programming is always extending a given system. Whereas modern languages, such as Modula, support the notion of extensibility in the procedural realm, the notion is less well established in the domain of data types. Modula in particular does not allow the definition of new data types as extensions of other, programmer-defined types in an adequate manner. An additional feature was called for, thereby giving rise to an extension of Modula.

The concept of the planned operating system also called for a highly dynamic, centralized storage management relying on the technique of garbage collection. Although Modula does not prevent the incorporation of a garbage collector in principle, its variant record feature constitutes a genuine obstacle. As the new facility for extending types would make the variant record feature superfluous, the removal of this stumbling block was a logical decision. This step, however gave rise to a restriction (subset) of Modula.

It soon became clear that the rule to concentrate on the essential and to eliminate inessential should not only be applied to the design of the new system, but equally stringently to the language in which the system is formulated. The application of the principle thus led from Modula to a new language. The adjective new, however, has to be understood in proper context. Oberon evolved from Modula by very few additions and several subtractions.

In relying on evolution rather than revolution we remain in the tradition of a long development that led from Algol to Pascal, then to Modula-2, and eventually to Oberon. The common trait of these languages are their procedural rather than functional model, and the strict typing of data. More fundamental even is perhaps the idea of abstraction: the language must be defined in terms of mathematical, abstract concepts without reference to any computing mechanism. Only if a language satisfies this criterion can it be called "higher-level". No syntactic coasting whatsoever can earn a language this attribute alone.

The definition of a language must be coherent and concise. This can only be achieved by a careful choice of the underlying abstractions an appropriate structure combining them. The language manual must be reasonably short, avoiding the explanation of individual cases derivable from the general rules. The power of a formalism must not be measured by the length of its description. To the contrary, an overly lengthy definition is a sure symptom of inadequacy. In this respect, not complexity but simplicity must be the goal.

In spite of its brevity, a description must be complete. Completeness is to be achieved within the framework of the chosen abstractions. Limitations imposed by particular implementations do not belong to a language definition proper. Examples of such restrictions are the maximum values of numbers, rounding and truncation errors in arithmetic, and actions taken when a program violates the stated rules. It should not be necessary to supplement a language definition with voluminous standards document to cover "unforeseen" cases.

But neither should a programming language be a mathematical theory only. It must be practical tool. This imposes certain limits on the terseness of the formalism. Several features of Oberon are superfluous from a purely theoretical point of view. They are nevertheless retained for practical reasons, either for programmers' convenience or to allow for efficient code generation without the necessity of complex, "optimizing" pattern matching algorithms in compilers. Examples of such features are the presence of several forms of repetitive statements, and of standard procedures such as INC, DEC, and ODD. They complicate neither the language conceptually nor the compiler to any significant degree.

These underlying premises must be kept in mind when comparing Oberon with other languages. Neither the language nor its defining document reach the ideal; but Oberon approximates these goals much better than its predecessors.

A compiler for Oberon bas been implemented for the NS32000 processor family and is embedded in the Oberon operating environment. The following data provide an estimate of the simplicity and efficiency of the implementation, and readers are encouraged to compare them with implementations of other languages. Measurements were made on a 10 MHz NS32032.

| area | lines | chars | bytes | seconds |
|---|---|---|---|---|
| Parser | 1116 | 3671 | 99928 | 11.53 |
| Scanner | 346 | 9863 | 3388 | 3.80 |
| Import/Export | 514 | 18386 | 4668 | 5.25 |
| Code generator | 19636 | 5901 | 21636 | 21.02 |
| Total | 3939 | 130869 | 39620 | 41.60 |

In the following is presented a brief introduction to Oberon assuming familiarity with Modula (or Pascal), concentrating on the added features and listing the eliminated ones. In order to be able to "start with a clean table", the latter are taken first.

Variant records are eliminated, because they constitute a genuine difficulty for the implementation of a reliable storage management system based on automatic garbage collection. The functionality of variant records is preserved by the introduction of extensible data types.

Opaque types cater to the concept of the abstract data type and information hiding. They are eliminated because again the concept is covered by the new facility of extended record types.

Enumeration types appear to be a simple enough feature to be uncontroversial. However, they defy extensibility over module boundaries. Either a facility to extend enumeration types would have to be introduced, or they would have to be dropped. A reason in favor of the latter, radical solution was the observation that in a growing number of programs the indiscriminate use of enumerations had led to a pompous style that contributed not to program clarity, but rather to verbosity. In connection with import and export, enumerations gave rise to the exceptional rule that import of a type identifier also causes the (automatic) import of all associated constant identifiers. This exceptional rule defies conceptual simplicity and causes unpleasant problems for the implementor.

Subrange types were introduced in Pascal (and adopted in Modula) for two reasons: (1) to indicate that a variable accepts a limited range of values of the base type and allow a compiler to generate appropriate guards for assignments, and (2) to allow a compiler to allocate the minimal storage space needed to store values of the indicated subrange. This appeared desirable in connection with packed records. Very few implementations have taken advantage of this space saving facility, because additional compiler complexity is very considerable. Reason 1 alone, however, did not appear to provide sufficient justification to retain the subrange facility in Oberon.

With the absence of enumeration and subrange types, the general possibility to define set types based on given element types appeared as redundant. Instead, a single, basic type SET is introduced, whose values are sets of integers from 0 to an implementation-defined maximum.

The basic type CARDINAL had been introduced in Modula-2 in order to allow address arithmetic with values from 0 to 2^16 on 16-bit computers. With the prevalence of 32-bit addresses in modem processors, the need for unsigned arithmetic has practically vanished, and therefore the type CARDINAL has been eliminated. With it, the bothersome incompatibilities of operands of types CARDINAL and INTEGER have disappeared.

The notion of a definable index type of arrays has also been abandoned. All indicies are by default integers. Furthermore, the lower bound is fixed to 0; array declarations specify a number of elements (length) rather than a pair of bounds. This break with a long standing tradition since Algol 60 demonstrates the principle of eliminating the inessential most clearly. The specification of an arbitrary lower bound provides no expressive power at all, but it introduces a non-negligible amount of hidden, computational effort. Only in the case of static declarations can it be delegated to the compiler.

Experience with Modula over the last eight years has shown that local modules were rarely used. The additional complexity of the compiler required to handle them, and the additional complications in the visibility rules of the language definition appear not to justify local modules.

The qualification of an imported object's identifier x by the exporting module's name M, viz. M.x can be circumvented in Modula by the use of the import clause FROM M IMPORT x. This facility has also been discarded. Experience in programming systems involving many modules has taught that the explicit qualification of each occurrence of x is actually preferable. A simplification of the compiler is a welcome side-effect.

The dual role of the main module in Modula is conceptually confusing. It constitutes a module in the sense of a package of data and procedures enclosed by a scope of visibility, and at the same time it constitutes a single procedure called the main program. Within the Oberon system, the notion of a main program has vanished. Instead, the system allows the user to activate any (exported, parameterless) procedure (called a command). Hence, the language excludes modules without explicit definition parts, and every module is defined in terms of a definition part and an implementation part (not definition module and implementation module).

The with statement has been discarded. Like in the case of exported identifiers, the explicit qualification of field identifiers is to be preferred.

The elimination of the for statement constitutes a break with another long standing tradition. The baroque mechanism in Algol 60's for statement had been trimmed considerably in Pascal (and Modula). Its marginal value in practice has led to its absence in Oberon.

Modula-2 makes access to machine-specific facilities possible through low-level constructs, such as the data types ADDRESS and WORD, absolute addressing of variables, and type casting functions. Most of them are packaged in a module called SYSTEM. The features were supposed to rarely used and easily visible trough the presence of SYSTEM in a module's import list. Experience has revealed, however, that a significant number of programmers import this module quite indiscriminately. A particulary seducing trap are Modula's type transfer functions.

It appears preferable to drop the pretense of portability of programs that import a "standard", yet system-specific module. Both the module SYSTEM and the type transfer functions are eliminated, and with them also the types ADDRESS and WORD. Individual implementors are free to provide system-dependent modules, but they do not belong to the general language definition. Their use then declares a program to be patently implementation-specific, and thereby non-portable.

The system Oberon does not require any language facilities for expressing concurrent processes. The pertinent, rudimentary features of Modula, in particular the coroutine, were therefore not retained. This exclusion is merely a reflection of our actual needs within the concrete project, but not on the general relevance of concurrency in programming.

The most important extension to Modula is the facility of extended record types. It permits the construction of new types on the basis of existing types, and establishing a certain degree of compatibility between the names of the new and old types. Assuming a given type

```
T = RECORD x, y: INTEGER END;
```

extensions may be defined which contain certain fields in addition to the existing ones. For example

```
T0 = RECORD (T) z: REAL END;

T1 = RECORD (T) w: LONGREAL END;
```

define types with fields x, y, z and x, y, w respectively. We define a type declared by

```
T' = RECORD (T) <field definitions> END;
```

to be a (direct) extension of T, and conversely T to be the (direct) base type of T'. Extended types may be extended again, giving rise to the following definitions:

A type T' is an extension of T, if T' = T or T' is a direct extension of an extension of T. Conversely, T is a base of T', if T = T' or T is the direct base type of a base type of T'. We denote this relationship by T' => T.

The rule of assignment compatibility states that values of an extended type are assignable to variables of their base types. For example, a record of type T0 can be assigned to a variable of the base type T. This assignment involves the fields x and y only, and in fact constitutes a projection of the value onto the space spanned by the base type.

It is important that an extended type may be declared in a module that imports the base type. In fact, this is probably the normal case.

This concept of extensible data type gains importance when extended to pointers. It is appropriate to say that a pointer type P' bound to T' extends a pointer type P, if P is bound to a base type T of T', and to extend the assignment rule to cover this case. It is now possible to form structures whose nodes are of different types, i.e. inhomogenious data structures. The inhomogeneity is automatically (and most sensibly) bounded by the fact that the nodes are linked by pointers of a common base type.

Typically, the pointer fields establishing the structure are contained in the base type T, and the procedures manipulating the structure are defined in the same (base) module as T. Individual extensions (variants) are defined in client modules together with procedures operating on nodes of the extended type. This scheme is in full accordance with the notion of system extensibility: new modules defining new extensions may be added to a system without requiring a change of the base modules, not even their recompilation.

As access to an individual node via a pointer bound to a base type provides a projected view of the node data only, a facility to widen the view is necessary. It depends on the possibility to determine the actual type of the referenced node.

This is achieved by a type test, a Boolean expression of the form

```
t IS T' or (p IS P')
```

If the test is affirmative, an assignment t' := t (t' of type T') or p' := p (p' of type P') should be possible. The static view of types, however, prohibits this. Note that both assignments violate the rule of assignment compatibility.

The desired statement is made possible by providing a type guard of the form

```
t' := t(T) or (p' := p(P))
```

and by the same token access to the field z of a T0 (see previous examples) is made possible by a type guard in the designator t(T0).z. Here the guard asserts that t is (currently) of type T0.

The declaration of extended record types, the type test, and the type guard are the only additional features introduced in this context. A more extensive discussion is provided in [2]. The concept is very similar to the class notion of Simula 67 [3], Smalltalk [4], and others. Differences lie in the fact that the class facility stipulates that all procedures applicable to objects of the class are defined together with the data declaration. It is awkward to be obliged to define a new class solely because a method (procedure) has been added or changed.

In Oberon, procedure (method) types rather than methods are connected with objects in the program text. The binding of actual methods (specific procedures) to objects (instances) is delayed until the program is executed. In Smalltalk, the compatibility rules between a class and its subclasses are confined to pointers, thereby intertwining the concept of access method and data type in an undesirable way. Here, the relationship between a type an its extensions is based on the established mathematical concept of projection.

In Modula, it is possible to declare a pointer type within an implementation module, and to export it as an opaque type by listing the same identifier in the corresponding definition module. The net effect is that the type is exported whereby its associated binding remains hidden (invisible to clients). In Oberon, this facility is generalized in the following way: Let a record type be defined in a certain implementation part, for example:

```
Viewer = RECORD width, height: INTEGER; x, y: INTEGER END;
```

In the corresponding definition part, a partial definition of the same type may be specified, for example

```
Viewer = RECORD width, height: INTEGER END;
```

with the effect that a partial view (a public projection) is visible to clients. In client modules as well as in the implementation part it is possible to define extensions of the base type (e.g. TextViewers or GraphViewers).

Modern processors feature arithmetic operations on several number formats. It is desirable to have all these formats reflected in the language as basic types.

Oberon features five of them:

```
LONGINT, INTEGER, SHORTINT (integer types)

LONGREAL, REAL (real types)
```

With the proliferation of basic types, a relaxation of compatibility rules between them becomes almost mandatory. Note that in Modula the arithmetic types INTEGER, CARDINAL and REAL are uncompatible. To this end, the notion of type inclusion is introduced: a type T includes a type T', if the values of T' are also values of type T.

Oberon postulates the following hierarchy:

LONGREAL > REAL > LONGINT > INTEGER > SHORTINT

The assignment rule is relaxed accordingly: A value of type T' can be assigned to a variable of type T, if T' is included in T (if T' extends T), i.e. if T > T' or T' => T. In this respect, we return to (and extend) the flexibility of Algol 60.

For example, given variables

```
i: INTEGER; k: LONGINT; x: REAL
```

the assignments

```
k:=i; x:=k; x:=1; k:=k+1; x:=x*10+i;
```

are confirming to the rules, where the assignments

```
i:=k; k:=x;
```

are not acceptable. Finally, it is worth noting that the various arithmetic types represent a limited set of subrange types.

The multi-dimensional open array and the closure statement (in symmetry to a module's initialization body) are the remaining facilities of Oberon not present in Modula.

The language Oberon has evolved from Modula-2 and incorporates the experiences of many years of programming in Modula. A significant number of features have been eliminated. They appear to have contributed more to language and compiler complexity than to genuine power and flexibility of expression. A small number of features have been added, the most significant one being the concept of type extension.

The evolution of a new language that is smaller, yet more powerful, than its ancestor is contrary to common practices and trends, but has inestimable advantages. Apart from simpler compilers, it results in a concise definition document [5], and indispensible prerequisite for any tool that must serve in the construction of sophisticated and reliable systems.

It is impossible to explicitly acknowledge all contributions of ideas that ultimately simmered down to what is now Oberon. Most came from the use or study of existing languages, such as Modula-2, Ada, Smalltalk, C++ and Cedar, which often though us how not to do it. Of particular value was the contribution of Oberon's first user, J. Gutknecht. The author is grateful for his insistence on the elimination of deadwood and on basing the remaining features on a sound mathematical foundation.

1. N. Wirth. Programming in Modula-2. Springer-Verlag, 1982.

2. N. Wirth. Type Extensions. ACM Transactions on Programming Languages and Systems 1988

3. G. Birtwistle, et al. Simula Begin. Auervach, 1973.

4. A. Goldberg, D. Robson. Smalltalk-80: The language and its implementation. Addison-Wesley, 1983.

5. N. Wirth. The Programming language Oberon

## MULTI-LANGUAGE LINKAGE

This discussion of multi-language linking was edited from a Usenet comment by Chris Torek at the University of Maryland Computer Science Department. Although the discussion is oriented toward unix, similar considerations would apply under other operating systems.

How can you link C, Fortran, and Pascal programs together under unix?

First, there is the matter of names: The symbols in the object files must match, so that the linker may resolve the right references. Each compiler has its own methods for mapping from source to object. Within one language we may usually ignore this mapping; but when mixing languages, it becomes important, as will be seen below.

The C compiler takes any global symbol and prepends an underscore character, '_'. Names are not limited in length; although in fact there is a limit of about a thousand characters, no one seems to be bothered by it.

Thus, the following fragment of code:

```
int global_var;

char *somefunc()
{

}
```

generates the symbols '_global_var' and '_somefunc'.

The Fortran 77 compiler limits names to six characters, then prepends and appends an underscore.

Thus, the following fragment of code:

```
subroutine sub
integer var
common /com/ var
```

names the subroutine '_sub_' and creates a global 'variable' containing one integer. The 'variable' is called '_com_'.

Variables that are not part of a common block do not have global names. Fortran 77 does not allow underscores in source-level names; 'subroutine sub_1' is illegal.

The Fortran 77 compiler also ignores any PROGRAM name, so that the following:

```
program prog

creates the symbol '_MAIN_'.
```

The Berkeley Pascal compiler strings together the names of all nested procedures to concoct unique global names. Only variables defined in the 'program' part are global (no surprise here), and these names are constructed in the same way as C's globals. However, the program name is ignored, and the compiler uses the name '_program'.

Thus, the following fragment of code:

```
program foo;              { symbol _program }
var v: integer;           { symbol _v }

procedure proc;           { symbol _proc }

function func;            { symbol _proc_func }
begin
func := 0
end;                      { end proc's func }

begin
end;                      { end proc }

begin
end.                      { end program }
```

generates the symbols '_program', '_v', '_proc', and '_proc_func'. It also generates the names '__proc_func' and '__proc', but they shall be ignored for the moment.

The Pascal compiler does not permit source-level names to contain '_'; thus, 'procedure proc_a' is illegal.

It should be clear at this point that C programs can call any Fortran 77 or Pascal subroutines (procedures) or functions, and that Pascal can call many C routines, but not all, for names with underscores are not directyl accessible, while Fortran 77 routines can call only specially-named C routines, namely those that end with an underscore, are less than seven other characters, and contain no internal underscores. Fortran 77 and Pascal routines can never call each other directly.

Even with a compatible set of names, the task is not yet done. There remain two problems, each bound up with the other. Every program must have an entry point ('main'); and every language has its libraries. C's is the simplest of the three, for its main looks like every other C routine and needs no libraries not used by both Fortran 77 and Pascal as well.

Fortran 77's main is actually a C-compatible routine which initializes its I/O system, traps signals, and calls the program's _MAIN_ function.

Pascal's main is similar to Fortran 77's, but does not trap signals and calls _program, not _MAIN_.

Both Fortran 77's and Pascal's mains also save argc and argv, Fortran 77's in _xargc and _xargv and Pascal's in __argc and __argv.

If you intend to call C routines from Fortran 77 or Pascal, and these routines are entirely self-contained, all that is necessary is to compile the C code to object, and mention the '.o' file in the linking command. Of course, you must also use the proper parameter passing conventions.

Calling Fortran 77 or Pascal routines from C, however, is somewhat more difficult. If the routines perform no I/O, they may simply be compiled from source to object and mentioned in the linking command. If they do I/O, you will need not only to initialize the I/O system, but also to clean up afterward. This becomes quite tricky and is best avoided whenever possible.

Fortran 77's support library is written almost entirely in C. Fortran 77's I/O system is initialized by the C routine 'f_init' and terminated by the routine 'f_exit'. Both take no parameters.

A Fortran 77 main program consists primarily of the following code:

```
f_init();
MAIN_(); /* recall that C prepends an underscore */
f_exit();
```

though there is much other code dealing with signals, and of course with argc and argv.

Pascal's I/O system is initialized by the 'PCSTART' routine, written in C.

Pascal's support library is also written in C. I find it amusing to note that other language libraries can be written in C, but C's language libraries cannot, for the most part, be written in the other languages.

Pascal's main can be written in C as the following:

```
extern int _argc;
extern char **_argv;

main(argc, argv)
int argc;
char **argv;
{


PCSTART(0);
_argc = argc;
_argv = argv;
program();
PCEXIT(0);

}
```

although the compiler in fact generates this directly, eliminating an unnecessary return instruction. PCEXIT, unfortunately, terminates the program as well as flushing any pending output.

As to the various libraries themselves, there are many, as follows:

| Library | Used by |
|---|---|
| -l77 | Fortran 77 |
| -lI77 | Fortran 77 |
| -lU77 | Fortran 77 |
| -lpc | Pascal |
| -lm | Fortran 77, Pascal |
| -lc | C, Fortran 77, Pascal |

In other words, all the linking commands pass '-lc' to the linker 'ld'; the others depend on the command. 'f77' calls ld with all except '-lpc'; 'pc' calls ld with '-lpc -lm -lc'. 'cc' calls ld with only '-lc', so to use an Fortran 77 routine with a C main, one must link with a command line similar to the following:

```
cc main.o f77sub.o -lf77 -lI77 -lU77 -lm
```

Moreover, the order of the libraries specified is also important. '-l77' builds on '-lI77', and '-lI77' builds on '-lU77'; all build on '-lm' and '-lc'. '-lpc' builds on '-lm' and '-lc'. Thus '-lpc' may be put anywhere with respect to '-lI77', for example; but both must appear before '-lm'.

From the preceding discussion, you should now be able to compile and link mixed-language source files together into one executable module.

However, this is not the whole story, as there is still the important, difficult, and confusing issue of how parameters are passed to functions in each of the languages.

The Fortran 77 compiler uses call by reference. The Pascal compiler uses call by value or call by reference, depending on the declaration of the called routine. The C compiler invariably uses call by value, but the language is powerful enough to simulate other parameter mechanisms using only call by value by passing addresses and pointers. One ability which can be accomplished in Pascal, but not C or in Fortran 77, is to pass arrays by value. This can be simulated in C using structures.

For strict definitions of call by value, call by reference, call by name, and other parameter-passing techniques and issues, consult a good compiler book.

Following are a few examples of mixed-language linking:

```
[f77sub.f]

SUBROUTINE SUB (A)
INTEGER A
DOUBLE PRECISION D
```

C Mixed mode arithmetic is legal in Unix Fortran 77:

```
D = A + 2.0
CALL CSUB(D)
RETURN
END


[psub.p]

{ declare external C subroutine }
procedure csub2(i: integer); external;

procedure psub(var i: integer);
begin i := 3 end;

function pfunc(i: integer): integer;
begin

pfunc := i + 2;
csub(i)

end;


[cmain.c]

main(argc, argv)
int argc;
char **argv;
{

int i;
psub(&i); /* call Pascal subroutine with var parameter */
sub_(&i); /* call Fortran 77 subroutine: call by reference */
i = pfunc(7); /* call Paacal function with value parameter */
exit(0);

}

/* called from Fortran 77: call by reference */
csub_(d) double *d; { printf("%g\n", *d); }

/* called from Pascal by value */
csub2(i) int i; { printf("%d\n", i); }
```

Fortunately, function return values are all done the same way for simple-valued functions. Structure-valued functions should simply be avoided.

Side-effects and the related issue of order of evaluation of arguments being evaluated for passing to functions present such difficult and implementation-dependent issues as to be avoided in all languages, at least for the purposes of this discussion.

Since the above example does no I/O in its Fortran 77 and Pascal routines, and in fact calls no Fortran 77 or Pascal intrinsics, this can be compiled with the following command lines:

```
f77 -c f77sub.f
pc -c psub.p
cc -c cmain.c
cc -o example cmain.o psub.o f77sub.o
```

Appending '-lF77 -lI77 -lU77 -lpc -lm' to the last command would not hurt, and might be required in more complex cases.

There is one remaining trick in linking Pascal and C or Fortran 77 routines, and that has to do with nested procedures and functions and nonlocal variable access. Neither C nor Fortran 77 have these, and there is no provision in the runtime environment for them. Pascal, however, uses something called a 'display' to be able to get at nonlocal variables. The display manipulation is normally compiled in-line; for procedure parameters, the compiler uses those 'extra' names. In the earlier example, these were '___proc' and '___proc_func'. These routines do display winding for entry to _proc and _proc_func. The unwinding after procedure parameter calls is generated in-line.

If you never use nested procedures, or nonlocal variables, you can safely ignore this. If you do, but do not know what a display is all about, again I will tell you only to consult a good compiler book.

Look at the assembly code generated by your Pascal compiler for details on the display format. Indeed, looking at the assembly code is a good way to determine just what the compiler is really doing for all three of these compilers.

+++

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™

# Basically OS-9

Dedicated to the serious OS-9 user.
The fastest growing users group world-wide!
6809 - 68020

*A Tutorial Series*

By: Ron Voigts
2024 Baldwin Court
Glendale Heights, IL 60139

## PUTTING DATA INTO MEMORY

Last column I left you with a program that created a menu driven environment that could be run on almost any OS-9 system. It presented menus that permitted access to more menus or ran programs. All the menus it used were to be located in a directory called /DD/MENU. I have not noticed if there are any major bugs, but I did come across an annoying problem. Every time a new menu is read, it is done by disk access. It takes time to find a file, open it, read it and close it. The result is menus appearing at whatever rate the file is read.

There are a few solutions. First, we can throw the whole thing away and pronounce it a failure. Second, get a hard disk with faster access time. Third, add a RAM disk and dump all the files to a reserved area in RAM. Finally, put it into memory in the form of a OS-9 module. We will pass on the first three ideas and go right to the last one. This month we will create a Data Module.

For late comers let me give the whirlwind tour of the OS-9 module. It is a special file that can be loaded into memory. It has a header, a body, and the CRC. The header identifies it as a loadable module. It has information regarding the name, module size, its type and language, its revision number and whether its sharable. The body is almost whatever is its intent. This could be executable object code, Basic09 I-Code, and Pascal P-Code. There is one other. It is the Data module.

The Data Module is not executable. It is only intended to hold information in memory. Putting the data into an OS-9 module means it can be loaded and linked into memory. This makes it a rather nice convenience. Key information used by a program can be kept in memory.

The header for the Data module is similar to other modules. It starts out with the usual two sync bytes $87CD. Then comes the module size and name offset. The type and language are $40. The attributes and revision are still used. The header has its parity. After the parity comes a entry location for the data and the amount of memory the module requires which is usually 0. The module header looks like this

| Address | Bytes | Usage |
|---|---|---|
| $00 | 2 | Sync Bytes ($87CD) |
| $02 | 2 | Module Size |
| $04 | 2 | Name Offset |
| $06 | 1 | Type/Language |
| $07 | 1 | Attribute/Revision |
| $08 | 1 | Header Parity |
| $09 | 2 | Data Offset |
| $B | 2 | Memory Size |

After the header comes the actual data. It can be whatever you want to store in memory. The OS-9 System has a few modules in the boot that are Data Modules. A familiar one is INIT which contains information used at startup. By the way, its Type/Language byte is $C0 for System Module/Data. Like all modules the last 3 bytes are the CRC for the module.

In Listing 1, I have a put a trivial example. This a sample assembly language program that contains information about the ownership of the computer. Actually its my address as it appears at the top of the column. This information may not be real important, but it does illustrate how to put data into a module form. A few points are of interest here. Notice that TYPE is set to DATA, this will become later $40. The symbol OwEnt is the data entry point. In an executable module, this would be where to start. Storage size is 0, so I merely put a 0 in the MOD statement.. The rest of the assembly is straight forward and similar to other assembly language listings for OS-9.

As I said this is a trivial example. It serves no purpose other than illustrative. But it is a good lead into Listing 2. This listing is for the menu program from last month. Remember when I mentioned earlier the long access time to the disk drive each time a new menu is displayed. Well, Listing 2 is one way to put the data in a form that can be loaded and saved in memory. The best way to use this one is to create all the menus, merge them and then load them into memory when appropriate. Probably, this would be done from the startup file.

Now let me explain a little further how this program works. This program uses two complex data types. One is for the menu entry and the other for the header. Some data types are declared including TITLE for the menu title, ESIZE for the number of entries, ENTRY is the complex variable for the entry and M is the complex variable for the header.

The header is similar to Listing 1. However, the values of M.SIZ and M.NAM are different. These are dependent on the length of the file ( the module ) name. Also, I set M.PAR equal to a value, but later it will get corrected.

The next part of the program creates the file with a temporary file name, inputs the necessary information and writes it. I varied from last months version slightly. The original version, 1.0, only wrote the entries that were required for the menu. This version writes all the entries. I set this value at 10.

To complete the file, a dummy 3 byte CRC number is written. This reserves space for the later CRC. Using Basic09's SHELL function the OS-9 command VERIFY is used to correct the header parity byte and the modules CRC. The temporary file is eliminated. The permanent module's attribute bytes — PE and E — are set using ATTR.

Once you have created the loadable data files, they can be put into memory with the LOAD command. A good idea would be to merge them under one file name. In OS-9 Level II, a module is put into a single block of memory. Each file loaded will be put into its own block. This is not very memory efficient. So by pre-merging, they are loaded successively into a single block.

All that is needed is to read them from the MENU program. The procedure in Listing 3 does this this method. It uses two techniques that differ from in Listing 2. Otherwise, the idea is the same, except it is reading the information rather than writing it. At this point in time, the modules have been loaded into memory.

The first technique is using a program called SYSCALL. The one I use is from the program package BASIC09 TOOLS available from SouthEast Media. It is also a part of the 3 Volume set of the programs from the Basically OS-9 column. These are also available from SouthEast Media. Another form of SYSCALL comes with newer releases of Basic09. If you use that one switch the arguments in the call. So on line 027B:

```
RUN syscall(regs,link)
```

becomes

```
RUN syscall(link,regs)
```

SYSCALL is used to access the OS-9 system calls F$Link and F$Unlink. F$Link links the module. The variable REGS consists of all the standard registers. REGS.X points to the module name and REGS.A is the type and language which is $40. The call returns with the REG.U pointing to the module's data entry address and REGS.Y pointing to its entry address. F$Unlink using REGS.U unlinks the module when finished. It is only passed the module's address, REGS.U.

Once the module's location is known, PEEK is used to transfer the data to the variables — TITLE, ESIZE, and ENTRY. In standard basic, PEEK is usually used to access a portion of memory. In OS-9 it is used the same, except that the module must first be linked into the processes memory area. Hence, the earlier use of F$Link and F$Unlink.

Listing 3 follows this order. Link to the module. Using PEEK transfer the data to the variable. And then Unlink the module. The call to the procedure is similar to the program Get_Menu from last month except I declare the parameter variables within GBet_Menu to be in bytes. This makes it easier to handle the data transfer.

## STYLO FOR COCO 3 OS-9

Before I leave this month, I must tell you that the column is once again being written with STYLO, the word processor from Stylo Software, Inc. The system I am running it on is the Coco 3 with OS-9 Level II. All I can say is it is great!

I could go on and tell you about its dynamic screen update feature. The screen shows how the printed page will look. I could tell about about how it will handle various printers. It will even do proportional spacing which makes my daisy wheel printer looks fine. In stead I will tell you about how it allows you to use the Coco 3 Windows.

Windows are the nice feature of allowing concurrent running process to occupy a part of the screen. Right now I have 3 shells running. The one I am on is an 80 column with green letters on a black background. Another is 80 column with yellow letters on black. And the third is a 40 column with blue letters on white. I can run stylo on 3 screens, if I wish.

Stylo for the Coco 3 permits the terminal to be configured for different types of screens. This is different from the older Coco version which held you to 3 fixed screen types — O-PAK, GO51 and Word-Pak. You are allowed up to 35 different terminal configurations. It comes with ones like RS Window 80 Column, RS Window 40 Column, DEC VT-52, and ADM-3A. You can also create new ones if you don't like these. Now you can attach an external terminal to the Coco 3 via an RS-232 port.

Next time I will tell more about windowing. For now let me say that STYLO is available from SouthEast Media for $69.95. This is a special price and I don't know how long it will last. If you want a fine word processor for your Coco 3 OS-9 Level II system, this is you chance.

That is it for now. See you next time!

Listing 1

```
      *******************************
      *
      * Name: Owner
      * By: Ron Voigts
      * Date: 12:DEC-1988
      *
      *******************************
      *
      * Version 1.0   Original
      *
      *******************************
      *
      * Purpose:
      *      This module shows how to
      *      create a loadable, data module.
      *      It is a simple example using my
      *      name, and address.
      *
      *******************************
      *
                           nam    Owner
      *
      * use /dd/defs/defsfile
                           ifpl
                           endc
      *
0040             TYPE      set    DATA
0081             REVS      set    REENT+1
      *
0000 87CD0050              mod    OwEnd,OwNam,TYPE,REVS,OwEnt,0
      *
000D 4F776E65    OwNam     fcs    "Owner"
0012 01          Version   fcb    1
      *
0013             OwEnt     equ    *
      * This the data area
0013 526F6E20              fcc    "Ron Voigts"
001D 0D                    fcb    C$CR
001E 32303234              fcc    "2024 Baldwind Court"
0031 0D                    fcb    C$CR
0032 476C656E              fcc    "Glendale Heights, IL 60139"
004C 0D                    fcb    C$CR
      *
004D F80A28                emod
      *
0050             OwEnd     equ    *
```

Listing 2

```
PROCEDURE make_menu
0000      (* **************************
001E      (*
0021      (* Name: Make_Menu
0033      (* By: Ron Voigts
0044      (* Date: 21-NOV-88
0056      (*
0059      (* ***************************
0077      (*
007A      (* Version 1.0         Original
0099      (*
 009C      (* Version 2.0         RDV
00B6      (*
00B9      (* *************************
00D6      (*
0009      (* This version will create a memory
00FD      (* module for the menu program.
011C      (*
011F      (* *************************
013C      (*
013F      (*
0142
0143      (* Set up complex data type
015E      TYPE entry_type=category:INTEGER; parameter:BOOLEAN; menu_line:STRING[64]; command:STRING[64]
01B5      TYPE module_header=snc:INTEGER; siz:INTEGER; nam:INTEGER; tl:BYTE; ar:BYTE; par:BYTE; ent:INTEGER;
mem:INTEGER
01BA
01BB      (* Set up variables
01CE      DIM s:STRING[32]
01DA      DIM t:STRING[1]
01E6      DIM path:BYTE
01ED      DIM title:STRING[64]
01F9      DIM esize:INTEGER
0200      DIM entry(10):entry_type
020E      DIM m:module_header
0217      DIM temporary:BYTE
021E      DIM version:BYTE
0225
0226      (* Get file name
0236      INPUT "Enter file name: ",s
024F
0250      (* Set  up constants
0264      m.snc:=$87CD
0270      m.siz:=$0571+LEN(s)
0281      m.nam:=$0D
028D      m.tl:=$40
0299      m.ar:=$81
02A5      m.par:=$58
02B1      m.ent:=$0E+LEN(s)
02C2      m.mem:=$00
02CE      version:=$01
02D6
02D7      (* Create a temporary file
02F1      CREATE #path,"temporary":WRITE
0305      PRINT
0307
0308      (* Write out header
031B      PUT #path,m
0325
0326      (* Write out name to file
033F      FOR i:=1 TO LEN(s)
0353        temporary:=ASC(MID$(s,i,1))
0363        IF i<LEN(s) THEN
0372          PUT #path,temporary
037C        ELSE
0380          temporary:=temporary+$80
038C          PUT #path,temporary
0396        ENDIF
0398      NEXT i
03A3
03A4      (* Write version
03B4      PUT #path,version
```

```
03BE
03BF         (* Get menu title
03D0         INPUT "Enter menu title: [64]: ",title
03F0         PUT #path,title
03FA         PRINT
03FC
03FD
03FE         (* How many entries?
0412         INPUT "Enter number of entries: [10]: ",esize
0439         PUT #path,esize
0443
0444         (* Get the the information for each item
046C         FOR i:=1 TO esize
047F           PRINT
0481           PRINT "Entry - ",i
0492           PRINT
0494           INPUT "Enter category: ",entry(i).category
04B4           PRINT
04B6           INPUT "Parameter?  (Y/N): ",t
04D1           IF LEFT$(t,1)="Y" OR LEFT$(t,1)="y" THEN
04EC             entry(i).parameter:=TRUE
04FA           ELSE
04FE             entry(i).parameter:=FALSE
050C           ENDIF
050E           PRINT
0510           INPUT "Enter menu line: [64]: ",entry(i).menu_line
0537           PRINT
0539           INPUT "Enter command: [32]: ",entry(i).command
055E         NEXT i
0569
056A         (* Write out the entry
0580         PUT #path,entry
058A
058B         (* Write out a dummy CRC
05A3         FOR i:=1 TO 3
05B5           PUT #path,temporary
05BF         NEXT i
05CA         CLOSE #path
05D0
05D1         (* Correct the CRC and header parity
05F5         SHELL "verify u <temporary >"+s
0612         SHELL "del temporary"
0623         SHELL "attr "+s+" e pe"
0638
0639         END
063B
```

Listing 3

```
PROCEDURE  get_menu
0000         (* ***********************
001A         (*
001D         (* Name: Get_Menu
002E         (* By: Ron Voigts
003F         (* Date: 13-DEC-1987
0053         (*
0056         (* ***********************
0070         (*
0073         (* Version 1.0  Original
008B         (*
008E         (* **********************
00A7         (*
00AA         (* This procedure links to a menu
00CB         (* in memory and passes the data
00EC         (* to the menu program.
0103         (*
0106         (* **********************
011F
0120         (* Passed arguments
0133         PARAM file(32):BYTE
013F         PARAM title(64):BYTE
014B         PARAM esize(2):BYTE
0157         PARAM entry(1310):BYTE
0163
```

```
0164        (* Set up complex data type
017F        TYPE registers=cc,a,b,dp:BYTE; x,y,u:INTEGER
01A4
01A5        (* Set up variables
01B8        DIM i:INTEGER
01BF        DIM regs:registers
01C8        DIM link:BYTE
01CF        DIM unlink:BYTE
0106
01D7        (* Constants
01E3        link:=$00
01EB        unlink:=$02
01F3
01F4        (* Fix file name with an EOL marker
0217        FOR i:=1 TO 32
0227          IF file(i)=$FF THEN file(i)=$00
0242         ENDIF
0244        NEXT i
024F
0250        (* Link to Module
0261        regs.a:=$40
0260        regs.x:=ADDR(file)
027B        RUN syscall(regs,link)
028A        IF LAND(regs.cc,$01)=1 THEN
0290          ERROR regs.b
02A5        ENDIF
02A7
02A8        (* Get menu title
02B9        i:=regs.y
02C4        count:=1
02CC        WHILE count<=SIZE(title) DO
02DC          title(count):=PEEK(i)
02EA          i:=i+1
02F5          count:=count+1
0301        ENDWHILE
0305
0306        (* Get menu esize
0317        count:=1
031F        WHILE count<=SIZE(esize) DO
032F          esize(count):=PEEK(i)
033D          i:=i+1
0348          count:=count+1
0354        ENDWHILE
0358
0359        (* Get menu entry
036A        count:=1
0372        WHILE count<=SIZE(entry) DO
0382          entry(count):=PEEK(i)
0390          i:=i+1
039B          count:=count+1
03A7        ENDWHILE
03AB
03AC        (* Unlink  Module
03BD        (* REGS.U should still point to module start
03E9        RUN syscall(regs,unlink)
03F8        IF LAND(regs.cc,$01)=1 THEN
040B          ERROR regs.b
0413        ENDIF
0415
0416        END
041B
```

+++

*FOR THOSE WHO NEED TO KNOW*

68 MICRO JOURNAL™

# SOFTWARE USER NOTES

A Tutorial Series

By : Ronald W Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

*From Basic Assembler to HLL's*

## Pat Enhancement

I have just returned from a visit to New Zealand with a few thoughts from John Spray regarding improvements in PAT. John suggested an "automatic bookmark" feature that would let you return to the last line that had been edited. To do that, I had to change two lines of PAT and add one more line. Now the sequence ESC ^G will get you to the last line that you edited, (i.e. actually changed something, typed somethng other than a command). I think it will be a handy addition to PAT, and I had never thought of the possibility before John mentioned it to me.
Thanks John.

John also thought it would be nice to allow a split screen and editing of two files at the same time, to facilitate moving portions of one file to another. I think that would be a nice addition too, but it will take a while to implement. Perhaps, it would be easier to use two different screens, one for each file being edited, and have a command to switch back and forth between files. That would be more straightforward, but it would still be a major undertaking in PAT. It would involve having two screen buffers, two edit buffers, two sets of pointers, etc. Perhaps I could swap values for the various pointers when switching screens. Anyway, it is a rather major undertaking, but it would add a feature that seems to be present in more and more of the latest editors. With this feature the user could load two files, mark a portion of one and move or copy it to the other, etc.

## Utilities for SK*DOS

While I was gone I received a pile of mail. One of the items was a disk from Michael Evenson in Texas. Michael sent me a bunch of utilities with documentation. He indicates that these are available on his BBS at 1-817-488-8398 and that they are also available on Peter Stark's BBS for SK*DOS. There were several sets of useful things on the disk, not the least interesting to me was a package called MSTOOLS. These contain utilities to read files from MS-DOS format disks to SK*DOS format. Included are MSDIR, which does a directory of an MS-DOS disk, MSREAD which reads a file from MS-DOS to SK*DOS, MSWRITE, which writes a file from SK*DOS to MS-DOS disks, and WRMSDOS which writes multiple files from SK*DOS to MS-DOS.

The documentation reads in part "This is a shareware product. You can have it for free or anyway you can get it. ... it's free but all donations are graciously accepted. If you don't contribute, don't complain about what it doesn't do. If you do contribute, you have the right to call my BBS and let me know what else you'd like it to do. If I think it's a grand idea, I'll add it and you'll get a diskette mailed to the address you gave me when you registered.... Donations for registration for MSTOOLS start at $25.00. That gives you one full year of attention at the complaint window."

I guess I'll send off my check for that package. It has already proven to be useful. I have just reconfigured a Tandy 1200-HD of my son's for a different printer. For some reason it didn't just come up and work right off. It is on my list of things to fix in the near future, but meanwhile my son David had a paper to do for school. He had started typing it in using PC-Write on the Tandy.

"How will I get this printed out?" "No problem", I said. I'll just copy it onto a 5" floppy, bring it down to my 68000 system and copy it onto the SK*DOS hard disk, insert a few formatting commands and print it out using JUST.

I hadn't tried MSREAD before, but I fired it up and it worked perfectly first try! I input the formatting commands and did a little proofreading and half an hour later we had 7 pages of text all nicely printed. Thanks Michael.

There is a larger package on the disk called CTOOLS. This one is a must if you do any "C" programming. You'll probably like and use it regardless of whether you are an occasional "C" programmer or you program in "C" all the time. The package includes:

FCHART:

A program to print a listing even if it requires multiple files to do it,

PP:

A Pretty Printing program that will print a source listing with proper indentation for compound statements etc.

CUTIL:

A general purpose filter program that can do case conversions, convert tabs to spaces, eliminate form feeds, make control codes visible, indicate 8th bit set, etc.

CBC:

This valuable utility checks for proper pairing of curly braces, parentheses, quotation marks, and comment delimiters in a program. I don't know about you, but I sometimes get comments mis-nested by leaving a close comment out. That results in a chunk of program that I expect to be compiled being considered a comment by the compiler. CBC catches and flags such omissions.

DIFF:

This is a file difference lister. Two files can be compared to see if they are the same. If not, DIFF can re-sync after an added line in one of the files is printed out. It truly does list the differences in two files.

GREP:

GREP is UNIXESE jargon for "FIND". You give it a string and a filename and it searches the file for the string, listing all lines in which a match occurs. GREP is an acronym for Global Regular Expression Parser. Such nonsense, had you not guessed, is in my opinion the ultimate in snobbery. I don't like unnecessary jargon in any field including my own (Electrical Engineering). In these times when one has to integrate mechanics, electronics, communications and computers, we don't need to make things unnecessarily difficult for someone outside of our particular field.

CCREF:

This program produces a nice cross-index of variables in a "C" program.

ASCII:

Prepares an ASCII code list for characters with codes from 0 to 127 decimal.

If you detect a little sarchasm with the name GREP, you are correct. It might as well be called FAST for Find Ascii String in Text file, or maybe LAOS for List All Occurrences of String. Somehow it really gets to me when programmers of operating systems go that far out of their way to come up with an obscure and meaningles name for a simple utility that could have a very descriptive name. My gripe is not with Michael Evanson, of course, but with the UNIX people who seem to thrive on the "let's make it hard for outsiders" attitude.

There were other programs on the disk as well, but the point of this is the availability of such utilities on bulletin boards for the taking if you have the proper communications software. Michael, how about a little article in '68' MJ about communications software. You had some on the disk you sent me. Tell the readers what they need to access your bulletin board and that of Peter Stark, to download these nice utilities. Are they (the utilities on the bulletin board) available in object form or only source???

### Assembler Utility

I have missed a clean-up utility for disk backup and other files and I had done a simple one called XBAK to delete backup files on a disk when along came the latest version of SK*DOS that has multiple directories. I realized that when I was in my T directory (Text files such as letters, etc.) I might want to type XBAK and have only the .BAK files in the current directory be deleted. I spent some time improving on XBAK. The result was a utility called XXX (for crossing out of multiple files). It works like this. Suppose you are in the T directory and you want to delete all the .BAK files. You simply command XXX BAK <CR> and all the files with the BAK extension (but only in the current T directory) are deleted. The utility assumes that you want to delete files on the WORKING drive. If you want to use XXX on your System drive you can temporarily make that your working drive also. For example if you use drive 0 for system, you can type WORK 0 to make 0 the working drive too. XXX is considerably more flexible than XBAK in that it allows you to define the extension of the files that you want to delete. Suppose for instance that you assemble your programs and output the listing to a .LST file to look at later. You might accumulate a number of .LST files in your assembler directory (I use directory A). Just type XXX LST and they will all be deleted. Previous to this implementation, I had several X utilities, namely XBAK, XCOM, XBIN, XLST, XOUT, etc. Of course each one deleted only files with the specific extensions.

Well, I thought about XXX for a while and decided the next day to add the capability of reading a filename with a wildcard character. I also decided that I could call the utility ZAP since it was fast approaching the usefulness of a utility by that name that had been supplied with the SWTPc version of FLEX for the 6809 for a long time. After a few tries I managed to get ZAP to read a filename from the command line and run through the current working directory looking for matches, deleting any file that matched. ZAP *.* would simply delete all files in a directory (just like the MS-DOS ERASE or DELETE utility). ZAP *.BAK would delete all backup files. ZAP FILE*.BAK would delete all files whose name starts with FILE and with the extension .BAK. The * doesn't limit the wildcard to one character, so that command would delete FILE1.BAK, FILETEST.BAK, FILE123.BAK, etc.

I was satisfied with that for a day, but I still wanted more. This command still couldn't handle the case of a different first letter such as in the series ACAT, TCAT, SCAT. Tonight I added a single character match wildcard, using a ? for that character. ZAP ?CAT.* would delete all of the three files mentioned above with any extension. I tested carefully to see that ZAP *.C would only delete files with the extension single letter C and not match .COM or .CMD. These can all be matched by using ZAP *.C*

At this point I am relatively satisfied and comfortable with the utility. The code is only 266 bytes and the listing just more than two pages. I'll include the listing here and make the source available on Peter Stark's bulletin board. I'll also send a copy to Sid Thompson so it will be available through the SK*DOS user's group in Atlanta.

Later - I did find another improvement for ZAP. As previously written, it had no provision for changing the case of the input command line. zap *.bak didn't do anything since my filenames are all upper case. I would have to use ZAP *.BAK. I added the code to convert the command line string to upper case and it works fine for my system. If you have set the flag so that filenames may be upper or lower case and be distinct and separate, you will want to commend out the lines that are indicated, that convert the command line parameter to upper case. If you are using upper case only filenames, just use the listing as printed. I've used this one frequently since completing it.

## Monochrome Monitor and the PT68K-2

The keyboard and monochrome board arrived from Peripheral Technology and I went out and bought a monitor locally, so I decided it was time to try PAT on the monitor and free up my serial terminal. Of course I found some more bugs in PAT and fixed them so now the operation is quite nice with that hardware. I had a silly bug that took me three evenings to track down. Once found it was totally obvious and simple to fix. I remarked that it looked like an unitialized variable and it was. Pat is about 53 pages of source listing so it took a while to figure out where to look for it.

I decided to do some playing and see if I could get the cursor control working. After a little detective work I found the combination and set PAT up so that in normal overlay mode the cursor is an underline but in insert mode it is a square, about half a full block cursor in height.

I am estimating the screen update rate to be about equivalent to a serial terminal running at 40K baud. That means that it can fill the 80 by 24 screen in just about 1/2 second when the whole thing is rewritten. I am having a little trouble getting used to the keyboard because the keys are a little soft for my touch, but they are relatively long stroke and they don't make until they bottom. The feel is a little indicative of a sudden "pop" to the made position as you push down on them. That could be called the "oilcan effect". My largest trouble seems to be that I drop spaces frequently if I am not paying attention. I think I'll like the keyboard a great deal after I use it a little more. It is one of the AT style keyboards and my ONLY complaint is that they stuck the ESC key over at the top right rather than at the left, next to the 1 key where it belongs. I haven't had all that much trouble getting used to its new location, however. It must be that I don't use the escape commands very often while entering text.

I found a 12 inch Packard Bell amber monitor with TTL inputs at the local Service Merchandise store. It was $89. Adding the price of the keyboard and monochrome graphics board I spent just about $205. I would have gone this way in the first place, except for the fact that I have a couple of serial terminals around. A nice serial terminal these days costs $400 to $500, so this is the relatively inexpensive way to go. I've spent some time working out the bugs in PAT that were revealed by trying to run it with the video board. Then, in response to a letter from a PAT user, I decided to tackle the displaying of marked blocks in the same attribute as the status line. With the monochrome board, I've used reverse video. I have the display working and I've tested most of the edit operations to see that they don't disturb the limits of the marked block. With the extent so clearly visible, I found a number of operations that caused the limits of the block to change needlessly, and fixed them. The monitor does not use a character space to change the attribute as some terminals do, so for a start, I have made the block marking work with so called non-modal terminals. The last step is to connect a serial terminal that is modal and make the marking work in that mode as well. I'll have that done shortly, but there will be a test version sent out to a few people who presently have PAT for SK*DOS. I'll expect a few funny situations to arise where there will have to be some fixes.

Having the full IBM style keyboard has prompted me to redefine some keys to make the cursor arrows and several other keys work as edit function keys. In addition to the arrows, I've defined HOME as goto top of file, END as Bottom of file, Pg Up to move up a page in the file, Pg Dn its obvious opposite function, INS and DEL to insert and delete blank lines respectively, and + and - to search forward and backward respectively if a search string has been defined. Since I use ^I for a tab key, I've also redefined the AT style keyboard TAB key (immediately to the left of Q) as an escape key, since that is where ESC is located on most terminals. I've left the original keys with their normal functions, so that, for example either ^Z or the keypad + key will search down, etc.

I plan to "retrofit" the 6809 version of PAT with this latest. (Of course I won't be able to implement the cursor pad key assignments). That version, incidentally now has quite a few added features over the last released version 2.5. If any of you would like an upgrade after I have finished debugging this latest change, send me a blank disk and a stamped return-addressed mailer in which to return your disk and I'll send you the PAT file, all the TERM files that I have, and a manual addendum describing the new features, along with the manual files for version 3.0. If you have only single sided double density disk drives, please send two disks. If you have single density and single sided please send three. In any event, please spell out the format that you want. I can accomodate 40 and 80 track drives, single or double density and single or double sided. This offer is for the FLEX 6809 version of PAT only. The SK*DOS version is not yet clean enough for an official release. Pending the completion of the present version and a test of the video version with the CGA adaptor and a color monitor, the SK*DOS version will be ready at last.

## Hardware Bug Cured

I have had the PT68k-2 system for some time now. When I wired up the parallel port connector on the back, I did it supposedly so I could put a DB-25 connector on one end of a 25 conductor ribbon cable and an Amphenol 36 contact connector on the other to be compatible with an Epson or other Centronics compatible printer. It just happened that I had some 34 conductor cable so I crimpped the Amphenol on the end of that. When I got to the other end, of course, I didn't have room for more than 25 connectors, so I separated the cable and cut off the 9 extra ones. What I didn't realize at the time is that I had left wires hanging on the Centronics end that became antennae for some of the printer control functions. The result was that when I printed a letter, I would sometimes get strange characters in place of spaces, and letters would be missed now and then. .p One day I happened to have an IBM compatible printer cable handy and I tried it. Much to my amazement, there were no errors. I looked at my ribbon cable and it dawned on me what I had done. I simply stripped the cable down to 25 conductors and cut the excess off at the printer connector. No more problems! The printer works fine with the supplied PARALLEL driver and the first 20 wires of the 25 pin connector. That is pins 1-10 and 14-23. Those are "left justified" in the amphenol connector so that pin 1 is pin 1. Connections are very straightforward. Pin 1 is the printer "strobe". Pins 2 to 9 are data bits 0 to 7, 0 being the low order bit. Pin 10 is the not acknowledge signal. The adjacent pins in the second connector row (19-28) are ground Pins 11 -13 may be left connected at the printer end since they are output signals from the printer. The offending wire in my cable was probably pin 31 of the amphenol connector at the printer. It is the not INIT signal which reinitializes the printer. Should you desire to make a universal cable for printers, the IBM standard is as follows:

Straight through indicates ribbon cable crimp-on connectors. Note that 1-13 of the DB-25 end up on 1-13 of the amphenol. Because the amphenol is two rows of 18 pins, the first pin on the bottom row would be pin 18,

DB-25 parallel port
Amphenol Printer Connector

| | | |
|---|---|---|
| 1-13 | straight through | 1-13 |
| 14 | (not straight through) | 14 |
| 15 | | 32 |
| 16 | | 31 |
| 17 | | 36 |
| 18-25 | straight through | 23-30 |

whereas the first pin on the bottom row of the DB=-25 is pin 14. The IBM standard would seem to use the first four pins on the bottom row of the DB-25 which would usually be ground pins, to connect four functions. Pin 14 is the signal for auto line-feed after CR, to the printer. Pin 15 of the DB to pin #@ of the Amphenol is the ERROR signal from the printer. Pin 16 of the DB to pin 31 of the Amphenol is the signal to initialize the printer. Pin 17 of the DB to pin 36 of the printer is the SELECT input of the printer. This input must be LOW to select the printer. I have purchased an IBM compatible printer cable that looked as though it simply had a DB-25 crimped on one end and an Amphenol 36 on the other. When I removed the cover at the Amphenol end, I discovered that the wires that would have been the first four in the bottom row were not connected there, but rerouted under the crimp cover to the pins indicated above and more or less hand crimped in place and dressed so that the cover would still fit. These four signals are not essential to the PARALLEL driver on the -2 (They are in fact not used). You could simply leave 14 and 15 open since they are signals FROM the printer. 16 should be connected to +5 volts (through a 1K resistor) and pin 17 can be ground along with 18-25.

Should you want to write a smarter printer driver that would report errors, initialize the printer, and check for printer selected, out of paper, or error, pin 11 is BUSY from the printer, pin 12 is OUT OF PAPER, and pin 13 is SELECTED. All positive (i.e. +4 to 5 volts) for TRUE.

+++

*FOR THOSE WHO NEED TO KNOW*
68 MICRO JOURNAL™

# Logically Speaking

Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you - *what you want!*

## The Mathematical Design of Digital Control Circuits

By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2

Copyrighted © by R. Jones & CPI

SOLUTIONS TO TEST F●URTEEN-B

| | | X | |
|---|---|---|---|
| | | 0 | 1 |
| 0 | 0 2 | 0 2 | 1 3 |
| 1 | 1 3 | 3 | 2 |
| 2 | 23 | 3 | - |
| 3 | 2 | - | 3 |

| | X | |
|---|---|---|
| | 0 | 1 |
| 0 | 02 | 1 |
| 1 | - | 2 |
| 2 | - | 3 |
| 3 | 3 | |

6 contacts
9 springs
(a)

| | | X | |
|---|---|---|---|
| | | 0 | 1 |
| 0 | 0 23 | 0 23 | 12 4 |
| 1 | 12 4 | 0 234 | 3 |
| 2 | 0 234 | 0 234 | 12 4 |
| 3 | 3 | - | 2 4 |
| 4 | 2 4 | 0 234 | - |

| | X | |
|---|---|---|
| | 0 | 1 |
| 0 | - | 1 |
| 1 | 3 | 3 |
| 2 | 023 | - |
| 3 | - | 24 |
| 4 | 4 | - |

7 contacts
12 springs
(b)

| | | X | |
|---|---|---|---|
| | | 0 | 1 |
| 0 | 0 23 | 023 | 1 3 |
| 1 | 1 3 | 023 | 2 |
| 2 | 2 | - | 3 |
| 3 | 3 | 023 | - |

| | X | |
|---|---|---|
| | 0 | 1 |
| 0 | - | 1 |
| 1 | 2 | 2 |
| 2 | | 3 |
| 3 | 023 | - |

5 contacts
8 springs
(c)

| | | X | |
|---|---|---|---|
| | | 0 | 1 |
| 0 | 01 | 01 | 12 |
| 1 | 12 | - | 123 |
| 2 | 123 | 3 | 123 |
| 3 | 3 | 3 | - |

| | X | |
|---|---|---|
| | 0 | 1 |
| 0 | 01 | 1 |
| 1 | - | 12 |
| 2 | - | 3 |
| 3 | 3 | - |

5 contacts
8 springs
(d)

|  |  | X |  |
|---|---|---|---|
|  |  | 0 | 1 |
| 0 | 0 | 0 | 1 |
| 1 | 1 | – | 2 |
| 2 | 2 | 23 | – |
| 3 | 23 | 23 | 4 |
| 4 | 4 | – | 5 |
| 5 | 5 | 5 | – |

|  | X |  |
|---|---|---|
|  | 0 | 1 |
| 0 | 0 | 1 |
| 1 | – | 2 |
| 2 | 23 | – |
| 3 | – | 4 |
| 4 | – | 5 |
| 5 | 5 | – |

8 contacts
13 springs
(e)

|  |  | X |  |
|---|---|---|---|
|  |  | 0 | 1 |
| 0 | 0 | 0 | 12 |
| 1 | 12 | 23 | 4 |
| 2 | 23 | 23 | 5 |
| 3 | 4 | 45 | – |
| 4 | 5 | – | 6 |
| 5 | 45 | 45 | 6 |
| 6 | 6 | 6 | – |

|  | X |  |
|---|---|---|
|  | 0 | 1 |
| 0 | 0 | 12 |
| 1 | – | 4 |
| 2 | 23 | – |
| 3 | – | 5 |
| 4 | 5 | – |
| 5 | – | 6 |
| 6 | 6 | – |

11 contacts
17 springs
(f)

|  |  | X |  |
|---|---|---|---|
|  |  | 0 | 1 |
| 0 | 0 23 56 | 023 56 | 01 3 5 7 |
| 1 | 01 3 5 7 | 023 56 | 012 5 8 |
| 2 | 012 5 8 | 023 56 | 0123 9 |
| 3 | 0123 9 | 4 789 | 0123 5 |
| 4 | 4 789 | 4 789 | 4 6 89 |
| 5 | 0123 5 | 023 56 | 0123 5 |
| 6 | 4 6 89 | 4 789 | 4 67 9 |
| 7 | 4 67 9 | 4 789 | 4 678 |
| 8 | 4 678 | – | 4 6789 |
| 9 | 4 6789 | 4 789 | 4 6789 |

|  | X |  |
|---|---|---|
|  | 0 | 1 |
| 0 | – | 01 |
| 1 | – | 2 |
| 2 | – | 3 |
| 3 | – | 5 |
| 4 | – | 46 |
| 5 | 02356 | 0 |
| 6 | – | 7 |
| 7 | – | 8 |
| 8 | – | 9 |
| 9 | 4789 | – |

18 contacts
28 springs
(g)

|  |  | X |  |
|---|---|---|---|
|  |  | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 1 | 1 | 2 | – |
| 2 | 2 | – | 2 |

Same table as original

4 contacts
6 springs
(h)

|  |  | X |  |
|---|---|---|---|
|  |  | 0 | 1 |
| 0 | 01 | 01 | 12 |
| 1 | 12 | 2 | 2 |
| 2 | 2 | 2 | - |

|  | X |  |
|---|---|---|
|  | 0 | 1 |
| 0 | 01 | 1 |
| 1 | - | 2 |
| 2 | 2 | - |

3 contacts
5 springs
(i)

|  |  | X |  |
|---|---|---|---|
|  |  | 0 | 1 |
| 0 | 0 23 | 023 | 01 45 |
| 1 | 01 45 | 5 8 | 013 7 |
| 2 | 5 8 | 5 8 | 9 |
| 3 | 01 3 7 | 67 9 | 013 5 |
| 4 | 9 | - | 10 |
| 5 | 67 9 | 67 9 | 6 8 10 |
| 6 | 01 3 5 | 5 8 | 013 5 |
| 7 | 10 | 10 | - |
| 8 | 6 8 10 | 10 | 6 8 9 |
| 9 | 6 8 9 | - | 6 8 9 10 |
| 10 | 6 8 9 10 | 10 | 6 8 9 10 |

|  | X |  |
|---|---|---|
|  | 0 | 1 |
| 0 | - | 01 |
| 1 | - | 3 |
| 2 | 023 | 0 |
| 3 | - | 5 |
| 4 | - | 7 |
| 5 | 58 | - |
| 6 | - | 68 |
| 7 | 67 9 | - |
| 8 | - | 9 |
| 9 | - | 10 |
| 10 | 10 | - |

17 contacts, 26 springs (j)

Mile 19 - heading for Mile 20

Last time we were together I mentioned that there's some jungle ahead, but it gets tougher so gradually I think maybe you should all keep going until it gets too tough, as you're bound to pick up some useful knowledge along the way. And who knows?, you may find that you can actually make it all the way through, especially as I'll be working hard to make the way as easy as possible. So let's give it a try, shall we? OK then, here we go into the territory of

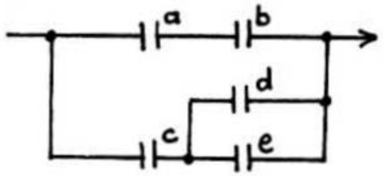
BOOLEAN MATRICES

CIRCUIT ANALYSIS

Prior to studying iterative networks, which are a very special branch of network theory, we've generally restricted the analysis and synthesis of our circuits to two-terminal networks of the series-parallel type, where power comes in at one terminal and goes out at another to operate some output device. At this stage of the game you should experience little or no difficulty in deriving the Boolean expression for any such network, or, alternatively, in constructing the network from the given Boolean expression. For example, if we look at the circuit of Diagram 99, we can very quickly write the Boolean expression as

$ab + c(d + e)$

or, if we started off with this expression, we can quite easily reconstruct the original network. Leastways, I sure hope so!



Continued on page 36

## PATH-TRACING

Now, how did we go about deriving the Boolean expression in the first place? Whether we really thought about it in this way or not, what we actually did was to write down the various paths which took us from input to output. That is, either (i) a AND b OR (ii) c AND (d OR e). This process is so commonly used that it's been given a name, appropriately enough it's called "path-tracing", and is a standard technique employed in order to write the Boolean expression for a network under consideration.

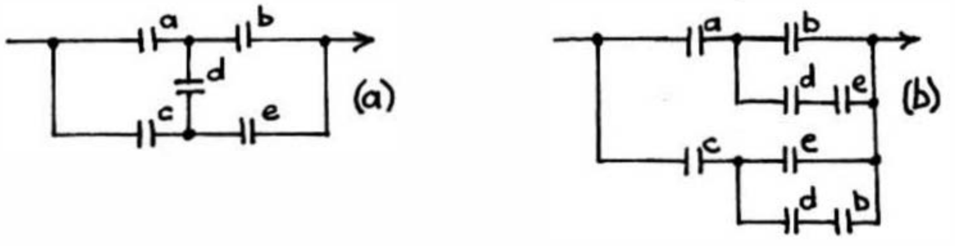


Diagram 100

We've already met the circuit of Diagram 100a in our discussion of bridge and non-planar networks, but we'll examine it now as a slightly modified version of Diagram 99. All we've done is to disconnect from the output one spring of the d-contact, and transfer it to the junction of the "a" and "b" contacts, but this simple move has severely complicated the derivation of its Boolean expression. We've previously puzzled over whether the left-most mesh should be written as "a + cd" or "c + ad". Or is it possible that neither is correct?
Path-tracing solves the problem for us! The Boolean expression for this circuit is really

ab + ce + ade + cdb

So far so good, but suppose that we were supplied with this expression to begin with, and asked to draw the circuit which it represents. This wouldn't SEEM to be a difficult task for us, as we'd merely factorise it to give (as one possibility) the expression

a(b + de) + c(e + db)

from which we'd draw the circuit of Diagram 100b. Although this circuit uses eight contacts instead of the original five, a machine wired in this way would be indistinguishable in operation from the first.

## FUNCTION v NETWORK

This discussion brings us to a most important point, namely that a Boolean expression gives us, NOT the circuit diagram, as we may have thought till now, but the conditions under which the network will transmit power. It describes the FUNCTION of the circuit (that is, the CONDITIONS under which an electrical path will be established), NOT the actual circuit itself nor its wiring!! As long as we restrict ourselves to series-parallel networks, we can read the expression as though it were also the circuit, but in the back of our minds we should always remember that THE FUNCTION IS NOT THE SAME AS THE CIRCUIT.
What we need, now that we're heading into more advanced systems, is some new way of describing a circuit mathematically, so that we're kept informed, not only of the function, but also the physical layout of the network. Fortunately for us, there is such a way, which makes use of the Boolean matrix (plural, matrices), which has the added advantage that we're able to keep track of several functions at one and the same time, PLUS all the intricate relationships which may exist among them.

## CONSTRUCTING A BOOLEAN MATRIX FROM A CIRCUIT-DIAGRAM

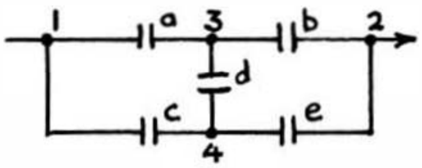


Diagram 101

The first step in constructing a Boolean matrix is to number all "nodes", or junction-points between contacts, as you see in Diagram 101. By convention, "1" is allocated to the power input-line, then the various output terminals are numbered, and finally all "non-terminal", or intermediate, nodes.

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1 & 0 & a & c \\ 2 & 0 & 1 & b & e \\ 3 & a & b & 1 & d \\ 4 & c & e & d & 1 \end{array}$$

Diagram 102

The Boolean matrix of Diagram 102 is a "fourth-order", or 4-by-4, matrix. Elements of the matrix are always referred to first by ROW, and then by COLUMN, *the reverse of our nonnal way of ieading a K-map! For example, a23 is the *element at the intersection of ROW-2 and COLUMN-3, and element a34 is the element at the intersection of ROW-3 and COLUMN-4, the "a" part signifying that we're looking at matrix-a, rather than matrix-b. The matrix is constuctcd from the network in the following manner :

On squared paper, we number a set of rows and columns from 1 to 4, to correspond with our nodes, and fill in the diagonal from top-left to bottom-*right with 1s. Then, in element a 12 we'll write the name of any contact which will transmit current from node-1 to node-2 WITHOUT GOING THROUGH ANY OTHER NODE. There's no such animal, so we insert "0" in this element. Next we'll *look at a13, and write here the name of any contact which will transmit *current from node-1 to node-3, which, of course, is the a-contact. In a14 we write "c", because this is the contact connecting nodes 1 and 4. OK so far? Glad you came along?

Now let's do Row-2, by commencing to the right of its 1-entry, that is, with **element a23, and insert here a "b", following on with element a24, where we insert an "e". Finally, in row-3, again commencing to the right of its 1-*element, we insert a "d" in element a34. And we're done!

We know that a relay contact is capable of passing current in either direction, which, in terms of our Boolean matrix, means that the path between nodes 2 and 3, for example, is the same as that between nodes 3 and 2. However, if we were considering a network in a DC (direct-current) system with diodes in some of the paths, this would not necessarily be the case, as current capable of flowing from node-2 to node-3 may be blocked by a diode if it tried to flow in the opposite direction. Having said that, let's get back to our matrix, where **we've decided in effect that element a23 IS the same as element a32, thus making our matrix symmetrical. Therefore we complete our filling-in by making column-1 read the same as row-1, column-2 the same as row-2, and so on. Remembering that "1" means a permanent connection, you'll readily see why the diagonals are filled in with 1s. It's because node-1 is obviously pennanently connected to node-1, node-2 to node-2, etc. Before we get down to studying this matrix a little more seriously, let's play around with it for a whi le and have some fun.

## CONSTRUCTING A CIRCUIT-DIAGRAM FROM A BOOLEAN MATRIX

First of all, I think we all agree that there'd be no problem in reconstructing the original network diagram from our matrix. We'd simply put down four dots on a sheet of paper, numbering the left-most with a "1", the right-most with a "2", and the intermediate ones "3" and "4". Then, reading only those elements above the diagonal (called "super-diagonal", as opposed to those below, which are "sub-diagonal"), we'd insert no connection directly between nodes 1 and 2, an a-contact between nodes 1 and 3, a c-contact between nodes 1 and 4, and so on and so forth. And there we'd have our original network, sometimes after a little straightening-out of lines!

## PATH-TRACING THROUGH A BOOLEAN MATRIX

Let's carry on "playing", shall we? Suppose we start on the figure "1" of row-1 outside the matrix proper, and then move horizontally through the row until we come under the "2" of column-2. Here we see the entry "0", which tells us that there is NO direct path from input to output. So let's try something different! Let's start again at the "1" of row-1 and move horizontally to, say, column-4, where we're informed that this IS possible via a c-contact, AND FURTHER that if we pivot off AT RIGHT-ANGLES down column-4 to row-2 we'll end up on an e-contact, which, of course, means that we can get from node-4 to node-2 via "e". We have, on this little trip, moved from node-1 to node-4 to node-2, that is, from input to output, through the contacts "c" and "e" (in that order). If we now look at our original network, we'll see that this is in fact a possible path from node-1 to node-2.

Suppose we try a different path through the matrix, each time doing a right-angled pivot from our current direction. Let's start

Telephone: (615) 842-4600

# South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

## ASSEMBLERS

ASTRUK09 from S.E. Media -- A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.
*FLEX, SK-DOS, CCF - $99.95*

**Macro Assembler for TSC** - The FLEX, SK-DOS STANDARD Assembler.
*Special -- CCF $35.00; FLEX, SK-DOS $50.00*

OSM Extended 6809 Macro Assembler from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. Generate OS-9 Memory modules under FLEX, SK-DOS.
*FLEX, SK-DOS, CCF, OS-9 $99.00*

**Relocating Assembler/Linking Loader** from TSC. -- Use with many of the C and Pascal Compilers.
*FLEX, SK-DOS, CCF $150.00*

MACE, by Graham Trott from Windrush Micro Systems -- Co-Resident Editor and Assembler; fast interactive A.L. Programming for small to medium-sized Programs.
*FLEX, SK-DOS, CCF - $75.00*

XMACE -- MACE w/Cross Assembler for 6800/1/2/3/8
*FLEX, SK-DOS, CCF - $98.00*

## DISASSEMBLERS

SUPER SLEUTH from Computer Systems Consultants Interactive Disassembler; extremely *POWERFUL!* Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.
*Color Computer SS-50 Bus (all w/ A.L. Source)*
*CCD (32K Req'd) Object Only $49.00*
*FLEX, SK-DOS $99.00 - CCF Object Only $50.00 UniFLEX $100.00*
*CCF, with Source $99.00 OS-9, $101.00 - CCO, Object Only $50.00*
*68010 SUPER SLEUTH - Similiar to 8-Bit Version except written in "C".*
*68010 Disassembler $100.00 FLEX, UniFLEX, UNIX, XENIX, MS-DOS, SK-DOS, OS-9*
*OS-9/68K Object Only $100.00 or with Source $200.00*

DYNAMITE+ -- Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options with OS-9 Version.
*CCF, Object Only $100.00 - CCO, Object Only$ 59.95*
*FLEX, SK-DOS , Object Only $100.00 - OS-9, Object Only$150.00*
*UniFLEX Object Only $300.00*

## CROSS ASSEMBLERS

CROSS ASSEMBLERS from Computer System Consultants -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/ 146805, 6809/00/01, 6502 family, 8080/5, 8020/1/2/35/C35/39/ 40/48/C48/49/C49/ 50/8748/ 49, 8031/51/8751,32000 and 68000/68010 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text. Includes Macro Pre-Processor. Written in "C". *68000 or 6809 *Macintosh,*Atari, FLEX, CCF, UniFLEX, OS-9, XENIX, UNIX, MS-DOS, SK-DOS*
*any object or source each -$50.00*
*any 3 object or source -$100.00*
*Set of ALL object $200.00 - with source $500.00*

XASM Cross Assemblers for FLEX, SK-DOS from S.E. MEDIA -- This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format, Assembler options, etc., in providing code for target CPU's.
*Complete set, FLEX, SK-DOS only - $150.00*

CRASMB from LLOYD I/O -- Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties); 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS-9 (binary), and FLEX, SK-DOS (binary). Written in Assembler ... e.g. Very Fast.

CPU TYPE - Price each:

| For: | MOTOROLA | INTEL | OTHER | COMPLETE SET |
|---|---|---|---|---|
| FLEX9 | $150 | $150 | $150 | $399 |
| SK-DOS | $150 | $150 | $150 | $399 |
| OS-9/6809 | $150 | $150 | $150 | $399 |
| OS-9/68K | ------ | ------ | ------ | $432 |

CRASMB 16.32 from LLOYD I/O -- Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS-9/68K on your OS-9/6809 computer.
*FLEX, SK-DOS, CCF, OS-9/6809 $249.00*

## COMMUNICATIONS

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".
*FLEX, SK-DOS, CCF, OS-9, UniFLEX, UNIX, XENIX, MS-DOS, with Source $100.00 - without Source $50.00*

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary files may be directly transfered between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.

| | |
|---|---|
| *X-TALK Complete (cable, 2 disks)* | *$99.95* |
| *X-TALK Software (2 disks only)* | *$69.95* |
| *X-TALK with CMODEM Source* | *$149.95* |

XDATA from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.
*UniFLEX- $299.99*

**Availability Legends**
**O = OS-9, S = SK*DOS**
**F = FLEX, U = UniFLEX**
**CC0 = Color Computer OS-9**
**CCF = Color Computer FLEX**

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

**** Shipping ****
**Add 2% U.S.A. (min. $2.50)**
**Foreign Surface Add 5%**
**Foreign Airmail Add 10%**
**Or C.O.D. Shipping Only**

***OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.**

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

## PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems -- By Graham Trott A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. 500+ page Manual with tutorial guide.

*FLEX, SK-DOS, CCF - $198.00*

PASC from S.E. Media - A FLEX9, SK-DOS Compiler with a definite Pascal "flavor". Anyone wit a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete wit three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package comes complete with source (written in PASC) and documentation.

*FLEX, SK-DOS $95.00*

WHIMSICAL from S.E. MEDIA Now supports *Real Numbers*. "Structured Programming" WITHOUT losing t e Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of t e Stack Pointer, etc. Run-Time subroutines inserted as called during compilation. *Normally produces 10% less code than PL/9.*

*FLEX, SK-DOS and CCF - $195.00*

KANSAS CITY BASIC from S.E. Media - *Basic for Color Computer OS-9* with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of t e usual string functions such as LEFT$, RIGHT$, MID$, STRING$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.

*CoCo OS-9 $39.95*

**C Compiler from Windrush Micro Systems** by James McCosh. Full C for FLEX, SK-DOS except bit-fields, including an Assembler. *Requires the TSC Relocating Assembler if user desires to implement his own Libraries.*

*FLEX, SK-DOS, CCF - $295.00*

**C Compiler from Introl** -- Full C except Doubles and Bit Fields, streamlined for the 809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible t an most.

*FLEX, SK-DOS, CCF, OS-9 (Level II ONLY), UniFLEX - $575.00*

**PASCAL Compiler from Lucidata** -- ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.

*FLEX, SK-DOS and CCF - $190.00*

**OmegaSoft PASCAL from Certified Software** -- Extended Pascal for systems and real-time programming.

Native 68000/68020 Compiler, $575 for base package, options available. For OS-9/68000 and PDOS host system.

809 Cross Compiler (OS-9/68000 host) $700 for complete package.

KBASIC - from S.E. MEDIA -- A "Native Code" BASIC Compiler whic is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.

*FLEX, SK-DOS, CCF, OS-9 Compiler /Assembler $99.00*

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSII Level 1 *COBOL* wit many of the useful Level 2 features. Full FLEX, SK-DOS File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run wit a single Disk System. *A very popular product.*

*FLEX, SK-DOS, CCF - $99.95*

**FORTH from Stearns** Electronics -- A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!

*Color Computer ONLY - $58.95*

FORTHBUILDER is a stand-alone target compiler (crosscompiler) for producing custom Forth systems and application programs.

All of t e 83-standard defining words and control structures are recognized by FORTHBUILDER.

FORTHBUILDER is designed to behave as much as possible like a resident Fort interpreter/compiler, so that most of the established techniques for writing Forth code can be used without change.

Like compilers for ot er languages, FORTHBUILDER can operate in "batch mode".

The compiler recognizes and emulates target names defined by CONSTANT or VARIABLE and is readily extended wit "compile-time" definitions to emulate specific target words.

FORTHBUILDER is supplied as an executable command file configured for a specific host system and target processor. Object code produced from the accompanying model source code is royalty-free to licensed users.

FLEX, CCF, SK-DOS - $99.95

## EDITORS & WORD PROCESSING

JUST from S.E. Media -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 wit Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line wid , margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use wit PAT or any other editor.

• Now supplied as a two disk set:

*Disk #1: JUST2.CMD object file,*
*JUST2.TXT PL9 source:FLEX, SK-DOS - CCF*
*Disk #2: JUSTSC object and source in C:*
*FLEX, SK-DOS, OS-9, CCF*

The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

**South East Media**

5900 Cassandra Smith Rd. · Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p .u .y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!

*Disk (1) - PL9 FLEX only- FLEX, SK-DOS & CCF - $49.95*
*Disk Set (2) - FLEX, SK-DOS & CCF & OS-9 (C version) - $69.95*
*OS-9 68K000 complete with Source - $79.95*

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.

*Regular FLEX, SK-DOS $129.50*
** SPECIAL INTRODUCTION OFFER * $79.95*
*SPECIAL PAT/JUST COMBO (with source)*
*FLEX, SK-DOS $99.95*
*OS-9 68K Version $229.00*
*SPECIAL PAT/JUST COMBO 68K $249.00*

Note: *JUST in "C" source available for OS-9*

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.

*FLEX, SK-DOS $69.95*

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entire lines, etc. Complete with over 25 different CRT terminal configuration overlays.

*FLEX, CCF, SK-DOS $39.95*

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

*6800 or 6809 FLEX, SK-DOS or SSB-DOS, OS-9 - $175.00*

SPELLB "Computer Dictionary" from S.E. Media -- OVER 150,000 words! *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX, SK-DOS UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

*FLEX, SK-DOS and CCF - $129.95*

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/SK-DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.

*NEW PRICES 6809 CCF and CCO - $99.95,*
*FLEX, SK-DOS or OS-9 - $179.95, UniFLEX- $299.95*

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.

*NEW PRICES 6809 CCF and CCO - $69.95,*
*FLEX, SK-DOS or OS-9 - $99.95, UniFLEX- $149.95*

STYLO-MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.

*NEW PRICES 6809 CCF and CCO - $59.95,*
*FLEX, SK-DOS or OS-9 - $79.95, UniFLEX- $129.95*

STYLO-PAK --- Graph + Spell + Merge Package Deal!!!

*FLEX, SK-DOS or OS-9 - $329.95, UniFLEX - $549.95*
*OS-9 68000 $695.00*

## *DATABASE ACCOUNTING*

**XDMS from Westchester Applied Business Systems**
**FOR 6809 FLEX or SK-DOS (5/8")**

Up to 32 groups/fields per record! Up to 12 character file names! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV

**Availability Legends**
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

**South East Media**
5900 Cassandra Smith Rd. · Hixson, Tn. 37343

master card

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

# South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

ITS EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...)

The possibilities are unlimited...

**FOR 6809 FLEX or SK-DOS(5"/8" Disk) $249.95**

## *UTILITIES*

Basic09 XRef from S.E. Media -- This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.

*OS-9 & CCO object only -- $39.95; with Source - $79.95*

BTree Routines - Complete set of routines to allow simple implementation of keyed files - *for your programs* - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.

*OS-9 & CCO object only - $89.95*

Lucidata PASCAL UTILITIES (Requires Pascal ver 3)

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.

*FLEX, SK-DOS, CCF --- EACH 5" - $40.00, 8" - $50.00*

DUB from S.E. Media -- A UniFLEX BASIC decompiler Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works with ALL Versions of 6809 UniFLEX basic.

*UniFLEX - $219.95*

LOW COST PROGRAM KITS from Southeast Media The following kits are available for FLEX, SK-DOS on either 5" or 8" Disk.

1. **BASIC TOOL-CHEST $29.95**
   BLISTER.CMD: pretty printer
   LINEXREF.BAS: line cross-referencer
   REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS:
   remove superfluous code
   STRIP.BAS: superfluous line-numbers stripper
2. **FLEX, SK-DOS UTILITIES KIT $39.99**
   CATS. CMD: alphabetically-sorted directory listing
   CATD.CMD: date-sorted directory listing
   COPYSORT.CMD: file copy, alphabetically
   COPYDATE.CMD: file copy, by date-order
   FILEDATE.CMD: change file creation date
   INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents
   RELINK.CMD (& RELINK82): re-orders fragmented free chain
   RESQ.CMD: undeletes (recovers) a deleted file
   SECTORS.CMD: show sector order in free chain
   XL.CMD: super text lister
3. **ASSEMBLERS/DISASSEMBLERS UTILITIES $39.95**
   LINEFEED.CMD: 'modularise' disassembler output
   MATH.CMD: decimal, hex, binary, octal conversions
   & tables
   SKIP.CMD: column stripper
4. **WORD - PROCESSOR SUPPORT UTILITIES $49.95**
   FULLSTOP.CMD: checks for capitalization
   BSTYCIT.BAS (.BAC): Stylo to dot-matrix printer
   NECPRINT.CMD: Stylo to dot-matrix printer filter code
5. **UTILITIES FOR INDEXING $49.95**
   MENU.BAS: selects required program from list below
   INDEX.BAC: word index
   PHRASES.BAC: phrase index
   CONTENT.BAC: table of contents
   INDXSORT.BAC: fast alphabetic sort routine
   FORMATER.BAC: produces a 2-column formatted index
   APPEND.BAC: append any number of files
   CHAR.BIN: line reader

BASIC09 TOOLS consist of 21 subroutines for Basic09.
6 were written in C Language and the remainder in assembly.
All the routines are compiled down to native machine code which makes them fast and compact.

1. CFILL -- fills a string with characters
2. DPEEK -- Double peek
3. DPOKE -- Double poke
4. FPOS -- Current file position
5. FSIZE -- File size
6. FTRIM -- removes leading spaces from a string
7. GETPR -- returns the current process ID
8. GETOPT -- gets 32 byte option section
9. GETUSR -- gets the user ID
10. GTIME -- gets the time
11. INSERT -- insert a string into another
12. LOWER -- converts a string into lowercase
13. READY -- Checks for available input
14. SETPRIOR -- changes a process priority
15. SETUSR -- changes the user ID
16. SETOPT -- set 32 byte option packet
17. STIME -- sets the time
18. SPACE -- adds spaces to a string
19. SWAP -- swaps any two variables
20. SYSCALL -- system call
21. UPPER -- converts a string to uppercase

For OS-9 - $44.95 - Includes Source Code

### SOFTOOLS

The following programs are included in object form for immediate application. PL/9 source code available for customization.

READ-ME Complete instructions for initial set-up and operation. Can even be printed out with the included text processor.

CONFIG one time system configuration.

CHANGE changes words, characters, etc. globally to any text type file.

CLEANTXT converts text files to standard FLEX, SK-DOS files.

COMMON compare two text files and reports differences.

COMPARE another check file that reports mis-matched lines.

CONCAT similar to FLEX, SK-DOS append but can also list files to screen.

DOCUMENT for PL/9 source files. Very useful in examining parameter passing aspects of procedures.

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC0 = Color Computer OS-9
CCF = Color Computer FLEX

VISA

**South East Media**
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

**ECHO** echos to either screen or file.

**FIND** an improved *find* command with "*pattern*" matching and wildcards. Very useful.

**HEX** dumps files in both hex and ASCII.

**INCLUDE** a file copy program that will accept "includes" of other disk files.

**KWIC** allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.

**LISTDIR** a directory listing program. Not super, but better than CAT.

**MEMSORT** a high-speed text file sorter. Up to 10 fields may be sorted. Very fast. Very useful.

**MULTICOL** width of page, number of columns may be specified. A MUST!

**PAGE** similar to LIST but allows for a page header, page width and depth. Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.

**REMOVE** a fast file deleter. Careful, no prompts issued. Zap, and its gone!

**SCREEN** a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.

**SORT** a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on n$^{th}$ word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.

**TPROC** a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.

**TRANSLIT** sorts a file by x keyfields. Checks for duplications. Up to 10 key files may be used.

**UNROTATE** used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.

**WC** a word count utility. Can count words, characters or lines.

**NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, with source (PL9). 3 5-1/4" disks or 1 8" disk without source.**

**Complete set SPECIAL INTRO PRICE:**
5-1/4" with source FLEX or SK-DOS - $129.95
without source - $79.95
8" with source - $79.95 - without source $49.95

**FULL SCREEN FORMS DISPLAY** from Computer Systems Consultants -- TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
*FLEX, SK-DOS and CCF, UniFLEX - $25.00, with Source - $50.00*

**SOLVE** from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No 'blind' debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!
*Levels I & II only - OS-9 $69.95*

## *DISK UTILITIES*

**OS-9 VDisk** from S.E. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.
*Level I OS-9 object $79.95; with Source $149.95*

**O-F** from S.E. Media -- Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK-DOS Format so it can be used normally by FLEX, SK-DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK-DOS Directory, Delete FLEX, SK-DOS Files, Copy both directions, etc. FLEX, SK-DOS users use the special disk just like any other FLEX, SK-DOS disk
*OS-9 - 6809/68000 $79.95*

**LSORT** from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.
*OS-9 $85.00*

**HIER** from S.E. Media - *HIER is a modern hierarchal storage system for users under FLEX, SK-DOS.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. Using HIER a regular (any) FLEX, SK-DOS disk (8 - 5 - hard disk) can have sub directories. By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. **Each directory looks to FLEX, SK-DOS like a regular file, except they have the extension '.DIR'.** A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK-DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!
*FLEX - SK-DOS $79.95*

**COPYMULT** from S.E. Media -- Copy LARGE Disks to several smaller disks. FLEX, SK-DOS utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.
*Completely documented Assembly Language Source files included. ALL 4 Programs (FLEX, SK-DOS, 8" or 5") $99.50*

**Availability Legends**
**O = OS-9, S = SK*DOS**
**F = FLEX, U = UniFLEX**
**CCO = Color Computer OS-9**
**CCF = Color Computer FLEX**

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

**•• Shipping ••**
**Add 2% U.S.A. (min. $2.50)**
**Foreign Surface Add 5%**
**Foreign Airmail Add 10%**
**Or C.O.D. Shipping Only**

***OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.**

Telephone: (615) 842-4600

# South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

COPYCAT from Lucidata -- *Pascal NOT required.* Allows reading TSC Mini-FLEX, SK-DOS, SSB-DOS68, and Digital Research CP/M Disks while operating under SK-DOS , FLEX1.O, FLEX 2.O, or FLEX 9.O with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.
*FLEX, SK-DOS and CCF 5".$50.00   FLEX, SK-DOS 8"-$65.00*

VIRTUAL TERMINAL from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eight tasks on one terminal, under *VIRTUAL TERMINAL* and switch back and forth between tasks at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.
6809 OS-9 & CCO - object only - $49.95

FLEX, SK-DOS DISK UTILITIES from Computer Systems Consultants -- Eight (8) different Assembly Language (with Source Code) FLEX, SK-DOS Utilities for every FLEX, SK-DOS Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten XBASIC Programs including: A BASIC Resequencer with EXTRAs over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.
*ALL Utilities include Source (either BASIC or A.L. Source Code).*
*FLEX, SK-DOS and CCF - $50.00*
*BASIC Utilities ONLY for UniFLEX -- $30.00*

MS-DOS to FLEX Transfer Utilities to OS-9 For 68XXX and CCOS-9 Systems Now READ - WRITE - DIR - DUMP - EXPLORE FLEX & MS-DOS Disk. These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks. *CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.
**CoCo Version: $69.95   68XXX Version $99.95*

## MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants -- TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.
*FLEX, SK-DOS and CCF, UniFLEX- $50.00, with Source - $100.00*

DYNACALC -- Electronic Spread Sheet for the 6809 and 68000.
*UniFLEX- $395.00, FLEX. SK-DOS. OS-9 and SPECIAL CCF - $250.00*
*OS-9 68K - $299.00*

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.
*FLEX, SK-DOS and CCF, UniFLEX - $50.00, with Source . $100.00*

FULL SCREEN MAILING LIST from Computer Systems Consultants -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.
*FLEX, SK-DOS and CCF, UniFLEX- $50.00, with Source - $100.00*

DIET-TRAC Forecaster from S.E. Media -- An XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.
*FLEX, SK-DOS - $59.95, UniFLEX - $89.95*

## GAMES

RAPIER - 6809 Chess Program from S.E. Media -- Requires FLEX, SK-DOS and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. This is one of the strongest CHESS programs running on any microcomputer, *estimated USCF Rating 1600+ (better than most 'club' players at higher levels)*
*FLEX, SK-DOS and CCF - $79.95*

*NEW*

*MS-DOS/FLEX Transfer Utilities* For 68XXX and CoCo* OS-9 Systems. Now Read, Write, DIR, Dump and Explore FLEX & MS-DOS Disks. Supplied with a rich set of options to explore and transfer text type files from/to FLEX and MS-DOS disks. *CoCo OS-9 requires SDISK utilities & two floppy drives.
CCO $69.95   68XXX OS-9 $99.95

## *MS-DOS and Macintosh Software at Discounted Prices*

**"Call for prices, it"ll be worth the savings."**

# *(615) 842-4600*

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. · Hixson, Tn. 37343

mastercard

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

again at row-1 and hop to column-4, then down to row-3 and across to column-2. This time we've journeyed through the sequence 1 - 4 - 3 - 2, and at the same time gone through the elements "c", "d" and "b" in that order. And what do you know? When we examine the actual network we find that this, too, is a possible path from node-1 to node-2.

We don't have to restrict ourselves to terminal-nodes! We could, for example, commence on row-3 and hop to column-4, which tells us that such a trip is possible from node-3 to node-4 via a d-contact. As an alternative, let's start on row-3, move to column-1, then down to row-4, where we learn that we can get from node-3 to node-4 by the sequence 3 - 1 - 4, corresponding to contacts "a" and "c" in series, and this is also true in the network itself. So you see, the Boolean matrix does give a very precise picture of the network's physical layout and wiring interconnections.

DERIVING A BOOLEAN FUNCTION FROM A BOOLEAN MATRIX

This is all very well, you say, but how does this matrix give me the Boolean FUNCTION of the network? It seems that a Boolean expression describes the FUNCTION but not the NETWORK, and a Boolean matrix describes the NETWORK but not the FUNCTION. Surely we can't be expected to go hopping around through the matrix, writing down EVERY possible path from node-1 to node-2 in order to derive the function! This might be possible with a small matrix such as this, but suppose it were a really huge one! How could we be sure that we hadn't overlooked a possible path here or there? Of course, we couldn't REALLY be sure, whether we were tracing all possible paths via a Boolean matrix or directly from a circuit-diagram either.

Not till now anyway, BUT, whereas such a task would be virtually impossible with the network itself, there ARE systematic methods for extracting a network's Boolean expression from a Boolean matrix. I'll describe two of the more common techniquesfor you, although there are other less commonly used methods. So here is

PIVOTAL CONDENSATION

The first technique is called "pivotal condensation", this method involving the successive elimination of non-terminal nodes until we're left with only those nodes of interest to us. Let's find the function f12 buried in our matrix (which will describe the transmission-function between nodes 1 and 2) by this system.

$$\begin{array}{c|ccc}
 & 1 & 2 & 3 \\
\hline
1 & 1 & ce & a + cd \\
2 & ce & 1 & b + ed \\
3 & a + cd & b + ed & 1
\end{array}$$

Diagram 103

We'll just adopt the minimal technique of eliminating one node at a time, though there are more sophisticated, but MUCH more difficult, ways of eliminating two or more at one time. So we'll begin by eliminating node-4, the highest-numbered, from the matrix of Diagram 102, though it's not as simple as merely crossing it out of the matrix. We must first draw up a new matrix (see Diagram 103), with both the rows and columns numbered for our reduced sequence 1 - 3, and the diagonals filled in with 1s. Just as before, we'll complete the super-diagonal elements one at a time, and finish off by making all columns read the same as their correspondingly numbered rows.

ELIMINATING A NODE FROM A BOOLEAN MATRIX

The first element requiring completion in our new matrix (let's call it matrix-*b) is element b12, so let's switch back to matrix-a in 102 and indicate that we've marked node-4 for elimination by drawing a red loop around all the elements in row-4, and also those in column-4, as though they were loops in a *K-map. Using element a12 as a pivot (now you see why it's called pivotal-condensation) we direct our attention horizontally to our circled column-4, where we see "c", and vertically downwards as well to row-4, where we see "e". Then we multiply these elements together (that is, we AND them) to give "ce", *and add the result (that is, we OR it) to the element a12, which is "0". This produces "0 + ce", which is logically equivalent to "ce", so this is what we *insert in element b12 in our new matrix-b in 103.

ONE DOWN - TWO TO GO

The next element to be completed is $b_{13}$, and so, back in matrix-a, we use $a_{13}$ as our pivot, and add to it the product of those elements occurring both horizontally and vertically in our circled column and row. We get "c" from column-4 and "d" from row-4, giving a product of "cd", which we add to the already-existing "a", to give "a + cd". This is duly recorded in element $b_{13}$!

## LAST ONE

Finally, using element $a_{23}$ as a pivot, we obtain "b + ed", which we enter in element $b_{23}$, and finish off by making the columns read the same as the rows.

## NOW WE HAVE MATRIX-B

What we've just done is to eliminate the FIGURE-4 from the network of Diagram 101, but leaving the circuit itself otherwise untouched. Now only nodes 1, 2 and 3 officially exist, and we see that matrix-b still describes this new netwoik. For example, commencing in row-1 and moving horizontally to column-3, this matrix informs us that we can now get from node-I to node-3 WITHOUT GOING THROUGH ANOTHER NUMBERED NODE (don't forget that node-4 no longer exists as a NUMBERED node) through "a + cd". That is, either via "a" OR via "cd", and so on with other paths, just as we did with matrix-a!

## DERIVING THE ACTUAL TRANSMISSION-FUNCTION

In order to find $f_{12}$, we still have to eliminate node-3, which would leave us with a 2-by-2, or second-order, matrix-c, with the single element $c_{12}$ to be filled in. In actual practice, once we've got down to a third-order matrix, we don't bother with creating the final matrix, as it's usually sufficient to calculate what the entry SHOULD be. Applying the same principles as before, we now pivot on element $b_{12}$ in matrix-b and add to it the product of the elements in column-3 and row-3, which are "a + cd" and "b + ed" respectively. We get

$$\begin{aligned} ce + (a + cd)(b + ed) &= ce + ab + aed + cdb + cde \\ &= ce + ab + aed + cdb \quad (\text{"ce" eliminates "cde"}) \end{aligned}$$

which is identical with the path found earlier by scanning the network itself. Although we know from the technique of pivotal condensation that the final expression above does in fact cover ALL possible paths through the network, IT DOESN'T GUARANTEE THAT THE CONTACTS OF EACH TERM ARE IN THE ORDER IN WHICH THEY OCCUR IN THE NETWORK. For instance, the third term in the Boolean expression is "aed", whereas the actual path is in the order "ade", but then, as we noted at the beginning of this part of our journey, the function isn't necessarily the same as the circuit itself. The FUNCTION of the term "aed" is to inform us that if relays A, E and D are operated, there will be a path from input to output. If we wish to know the ORDER in which these contacts appear in the path, we'll have to go back to the circuit-diagram or to the original matrix-a, and do a little path-tracing in either case!

Nodes may be eliminated in any order, the procedure remaining unchanged. We could, for instance, have begun by circling both row-3 and column-3 in red, and eliminated node-3 first, but it's usual to move inwards from the outer edge of each succeeding matrix.

## TIME TO SET UP CAMP

And here's where I think we'll pitch our tents for the night, and let all this sink in! I hope you didn't find it as frightening as you thought it might be at first, and that it's given you an interestingly new viewpoint on the relationship between function and network, perhaps something you've never really thought about till now. If any of you had difficulty in following my explanation, now's your chance to let me know, and maybe I can expand, or expound, further on Pivotal Condensation. Next time we'll look at "Laplace's Development", which, I have to admit, is a LITTLE trickier than our current friend P.C. (nothing at all to do with Personal Computer, let me add).

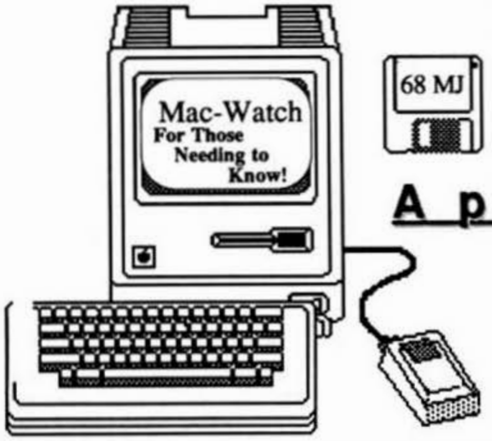
And as you've all struggled so willingly to keep up with me, I haven't the heart to give you a TEST right now. So just rest easy, and when we've mastered the next stage I'll give you a combined TEST on both techniques. Fair enough?

... End of Mile 19. Now at marker Mile-20. Roughly half-way through our trip, I'd guess, give or take a mile or two, or maybe three!

+++

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™



*The Macintosh™ Section*

Reserved as

**A place_for_your thoughts**

*And ours.......*

**Mac-Watch**

# A Review of StandingOut

**Desktop Presentation Software**
**from Lectrset, Inc.**

By James E. Law
1806 Rock Bluff Rd.
Hixson TN 37343

When PageMaker burst on the Macintosh scene, few could have predicted that this single event would be so important in setting off the development of something called desktop publishing. A new niche market was created which probably had more to do with leading to corporate acceptance of the Macintosh than any other application. Since then, an entire industry has sprung up to support DTP involving DTP magazines, DTP service bureaus, DTP specialized software, etc. DTP has made a lot of money for a lot of people.

It's not surprising then, that industry insiders would like to repeat this profitable blip in the market by developing other niche markets. That's why you hear of desktop this and desktop that. One of the latest areas to be so labeled is desktop presentations.

Presentations are a major part of the American business environment. If you want to sell products or ideas, presentations are almost always involved. Also, in reporting progress or status, in search for a group consensus, or in collegial reviews, presentations are often involved. According to Success Magazine (Jan-Feb 1989), there are 25,000,000 such presentations in the United States every day!

Presentations vary from informal, 'sit around the table' affairs to elaborate multimedia shows. Either can be effective, but researchers have discovered that presentations accompanied by visuals are more likely to be successful. In a much quoted 1986 study by the University of Minnesota, researchers found that presentations with visuals are up to 43% more effective than those in which a speaker just talks. Also, the speaker's credibility can be affected by the quality of his visuals. I remember a formal presentation in my company where highly paid consultants were to present the findings of a 3-month study. When black and white transparencies made from a mixture of typed and handwritten pages were used, my perception of the consultants' competence sagged.

In the pre-desktop presentations era, the options for generating visuals were limited. Often an overhead transparency for a typed page was the extent of it. Of course, if you had the funds to hire a graphics agency or were big enough to have graphics department in-house, then fancier presentation visuals were possible.

Thanks to a new generation of software that made its debut on the Macintosh, anyone can quickly prepare high quality visuals. Of course, you still have to have some graphics knowledge to turn out works of art, but anyone with a few hours training can turn out colorful text charts with graphics elements.

Over the last year, the number of such software packages has expanded to include More II, Power Point, Cricket Presents, StandingOut and others. I highly doubt that the desktop presentation market is big enough to support all these contenders. A large company may need many hundreds of copies of word processing software or data base software but how many desktop presentation programs will they need? Only a few, at best.

In this review, I would like to look at StandingOut which is distributed by Letraset, Inc. StandingOut was originally distributed under the name ReadySetShow. Apparently this detracted from Letraset's other major package ReadySetGo, so the name was changed.

As regular readers of 68 Micro Journal will remember, a detailed review of ReadySetGo was published in 1987. Later, a follow up revision to this fine program was written up. If you are familiar with ReadySetGo, then you know almost everything you need to know about StandingOut. They probably share 70% of the same coding. As a matter of fact, if you take ReadySetGo, remove the ability to work in columns, and add the ability to sort slides and create auxiliary documents (e.g., notes pages, and handout pages), you have StandingOut. This is not to say that StandingOut is deficient in power. As we will see, this sharing of the best of its cousin ReadySetGo, well equips StandingOut to produce stunning results.

StandingOut basically provides 3 functions. It allows you to design attractive and effective visuals in 35mm, overhead transparency, or other formats with a variety of text and graphics elements. It provides for the easy sorting of these visuals and of presenting them in an on-screen slide show. Finally, StandingOut allows you to easily prepare notes pages or audience handouts containing your visuals. Let's start by examining StandingOut's ability to work with text.

### Words, Words, and More Words

You can do anything with text in StandingOut that you can do with most desktop publishing programs. The spacing of paragraphs, lines, and words can be adjusted. Letter spacing and kerning are supported. Automatic hyphenation keeps your text lines straight. A spelling checker with an 85,000 word dictionary ensures speakers don't lose credibility through spelling errors in their visuals. Of course, tabs, glossaries, and almost all functions of a full featured word processor are provided. Reverse type is supported. You can easily cause text to wrap around regular or irregular objects.

Other extras are provided which add a nice touch to text slides. For example, text entries may automatically framed, filled, and/or shadowed. Also, leading or trailing characters may be automatically added to text entries. For example, a 'bullet' symbol can be made to proceed each line in a text bullet chart and will automatically appear when RETURN is pressed. Another helpful 'extra' is a option that allows you to change the case of selected text. You can make the selected text all capitals, all lower case, first letter of each word capitalized, or the first letter of sentences capitalized.

### High Style

What happens when you finish a presentation using Geneva fonts then decide that the headings should be Cooper Black and sub tier bullets should be Helvetica bold? With some presentation packages, a lot of reword would be required. Not so with StandingOut if you used style-sheets. A style sheet is simply a set of formatting instructions which you name and save. To set up a style sheet you select style from the Presentations menu, indicate that you want to set up a "New" style sheet, give a name to your style sheet, then enter the desired specifications for your text. You may set the font, size, style, alignment, indents, spacing (word, line, and paragraph) and hyphenation.

When you get ready to type text, you may then select an existing style sheet and the text will be formatted accordingly. If all the headings in your presentation were formatted under the control of the same style sheet, you can reformat all the headings at once by changing the Style Specification sheet. StandingOut allows you not only to modify and use style sheets, but also to import them from other presentations and to duplicate or delete them.

### Fancying It Up

No one buys a high powered desktop presentation program just to place text on a visual. To make visuals interesting and effective, graphics elements are usually added. An extra benefit is that consistent graphics elements (e.g., colors, borders) used throughout a presentation tend to tie the presentation together—to add visual continuity.

StandingOut provides a full list of graphics tools which can be used to prepare any "draw" type image you may need. The resulting objects may be sent to the front (or back), grouped (or ungrouped) aligned, duplicated, locked in place, printed or not printed, and otherwise manipulated. By selecting "Special Effects," you can automatically add a 'shadow' to objects. Specification boxes may be called up for each object to precisely specify the object's position by screen coordinates. In addition to the usual geometric shapes (e.g., ovals/circles, rectangles, lines), StandingOut allows you to draw a variety of parallelograms and triangles. While bit mapped images can be imported into StandingOut, this software contains no tools to develop or modify bit mapped images.

StandingOut has extensive charting capability. A wide variety of line, bar, pie, and other charts may be produced with significant flexibility to customize the final product.

### Color

No black and white presentation will ever attract as much initial attention as one in living color. StandingOut provides you with 256 premixed colors. Additionally, you can customize your own colors and even give names to your custom colors. You can assign a color to text, any other object on your slide, and to the background of the slide. You can assign colors even if you are working with a monochrome screen. Although all you will see is black, white, and gray, the assigned colors will be saved with your file and can be viewed later if you have access to a color monitor.

### Slide Design System

Have you ever designed a slide that you were especially proud of—that was especially well balanced and artistic? Perhaps you may want to do another slide in a future presentation similar to the one that worked so well last time. SteppingOut facilitates the saving and reuse of designs in a feature called the Slide Design System. By selecting "Add Design" from the File menu, the general design of the slide currently displayed is added to a library of saved designs. This saved design, which you can name, contains graphics features, text blocks, and picture blocks. Text and picture blocks are empty and do not maintain their contents. The format of text (size, font, style, color) to be included in text blocks is maintained, however.

### Continuity of Design

When you start up StandingOut you can use Page setup to indicate what format your visual will take (35mm, overhead transparency, or other). Oddly, StandingOut assumes that overhead transparencies will be in a vertical format and if you want a horizontal format, you will have to enter the appropriate dimensions. (I would have thought that most overhead transparencies are in horizontal format, not vertical, and the default specifications would have been set accordingly.)

"Master" templets can be set up for a presentation such that selected text or graphics elements entered on the master will appear on each visual in the presentation. You can use this feature, for example, to place a logo and border in exactly the same place on each visual. This not only saves time, but adds cohesiveness, continuity and professionalism to your presentation.

### Order Out of Chaos

If you are like me, you finalize your presentation by choosing slides from previous presentations, drawing some new slides, culling out the less effective ones, and rearranging the order of the slides several times. StandingOut lets you accomplish these functions painlessly. Slides can easily be transferred between files (presentations). You can view thumbnail size images of your slides on the screen and rearrange their order merely by clicking on a slide and drag it to where you want it. Slides may be deleted by selecting the thumbnail image and pushing BACKSPACE. You can also view a list of the titles of your slides and use this window to delete slides or rearrange their order as above.

### Notes Pages and Audience Handout Pages

StandingOut, like other desktop presentation packages, provides for easy preparation of notes pages and audience handout pages. Notes pages contain a miniature image of a slide, any standard features you want (e.g., titles, dates, logo, border), and your speaking notes. The audience handout pages contain either 2, 4, or 8 miniature slide images on a page along with any desired standard features. This certainly saves on copying cost and provides the audience a concise reminder of what you had to say.

### Show Time!

StandingOut can present the visuals of a presentation on screen as a slide show. You can set the number of seconds that each visual will be displayed or can program the visuals to change when the mouse is clicked. Additionally, a variety of transitions can be used in going from one visual to another. (e.g., wipe left to right, fade, etc.) The slide show can also be controlled from the keyboard. This is a very handy capability. You can gauge the overall effect or your presentation by viewing it as a slide show. Nonstandard slides will stand out like a sore thumb. If you have access to a LCD screen projector, you can use the slide show feature to present you graphics directly from the computer to your audience without having to prepare transparencies or slides.

### Just Read the Manual

StandingOut comes with a general users manual and a collection of minor manuals covering the tutorial and the design of presentation graphics. The general users' manual is a quality product and does a complete job of clearly explaining the use of StandingOut. There is a major defect, I believe, in the overall documentation package and that is the lack of effective examples of what you can do with this product. Don't expect to get a lot of good ideas about the design and layout of slides from StandingOut's manuals. Instead, you will find simplistic and unattractive black and white examples of visuals. Even the templets provided with StandingOut strike me as being remarkably unimpressive. The inclusion of well designed examples (preferably in color) would be of tremendous help to users and no doubt would help to sell this package.

### The StandingOut Interface

StandingOut shares essentially the same interface as its look alike ReadySetGo. I find it to be easy enough to use, but some users find it a little more hostile environment than some other related programs. For example, before you can enter text, you must draw a text block. Before you can import graphics, you must draw a graphics box, then select a different tool to activate the box to receive the graphics. I would have to agree that StandingOut is not as simple as PowerPoint, but on the other hand, it's a great deal more powerful so, pay your money and make your choice.

### Does StandingOut Stand Out?

This product is an excellent performer that should provide all the power that most users will ever need in preparing visuals. It has far more features that other similarprograms that I have used yet is priced quite competitively when compared with other top-end desktop presentation programs. If you already use ReadySetGo, and need a desktop presentation program, then I recommend you give special consideration to StandingOut since you will have already mastered 70% of its features. Is StandingOut the program for you? I can't answer that for you but I can say that this package does what it is advertised to do and represents a solid value.

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™

# C'ing LOGIC

By: J.A.Woodward
556 W. Main St.
Lock Haven, PA 17745
717 748 8837

This article will attempt to explain propositional logic and discuss the accompanying program which produces Truth Tables as they appear in most elementary logic texts. Programmers need to know a little formal logic, as it is used in almost every program written.

We will take as undefined terms the following: 2t2r2u2e and 2f2a2l2s2e. A statement is defined as a declarative sentence which is either true or false, but not both. A statement variable is a symbol that represents the truth value of a statement. The program allows the use of the letters p, q, r, and s as statement variables, so in this article I shall use only these letters to represent statement variables.

The following examples are statements:

1) 1+1=2
2) George Washington was a president of the U.S.A..
3) Everyone hates mathematics.
4) Tennessee is larger than Texas.

The following examples are not statements:

5) George Washington was a good president.
6) Why am I reading this?
7) Country music is better than heavy metal.
8) 2+x=5

1) and 2) are under normal circumstances considered true statements while 3) and 4) are considered false statements. I like mathematics and this fact is sufficient to make 3) false. 5) and 7) are not statements because they represent opinions, not facts. 6) is an interrogative sentence, not a declarative sentence and 8) has an unknown. Sentences as 8) are the basis for Predicate Logic, another type of logic.

Some statements are called compound statements, this means that they contain at least one connective. There are two types of connectives,unary and binary. We shall discuss one unary connective, called negation (not), and four binary connectives, called conjunction (and), disjunction (or), conditional (if,... then), and biconditional (...if and only if...). The word or words in the parentheses are the common everyday words usually used as connectives. The word, iff (some authors use iffi), is usually used as a shortened form of ...if and only if... . A statement that contains no connective is called atomic. Some examples of compound statements are:

9) 1+1=2 or the moon is made of blue cheese.
10) If 2+3=6, then 2+2=4 .
11) Ronald Reagan is not the President of the U.S.A. .

If the above are statements, then they must be either true or false. How does one decide the truthfulness of a compound statement? The answer to that question is: by definitions. A statement variable is a symbol which represents true (T) or false (F). True or false are called values. We shall use the symbols &,V,>, and = to represent the binary connectives conjunction, disjunction, conditional, and biconditional respectively. The symbol, ~, is used to represent negation. Each compound statement can be symbolized by using statement variables to represent the atomic statements, and ~,=,>,V, and & to represent the connectives. Some examples of symbolization are as follows. Let p represent: 1+1=2, and q represent: The moon is made of blue cheese., then statement 9) above is symbolized by: pVq. Statement 10) could be symbolized as follows: let s represent: 2+3=6 and r represent: 2+2=4, then statement 10) is symbolized by: s>r . Statement 11) can be symbolized by; ~q where q represents the atomic statement: Ronald Reagan is the president of the U.S.A. . An easy way to define the connectives is by using tables (ones with no legs). The following table defines the connective &:

```
& | T F
—
T | T F
F | F F
```

Since the symbol, =, is used to represent the connective biconditional, I shall use 2= to represent the equals concept. With that in mind, the connective conjunction (&) could also be defined by the following equations:

```
1)  T & T 2= T
2)  T & F 2= F
3)  F & T 2= F
4)  F & F 2= F
```

This means that a compound statement which is the conjunction of two statements (called arguments) is true when and only when both arguments are true. The connective disjunction is defined by the following table:

```
V | T F
—
T | T T
F | T F
```

or by the following equations:

```
1)  T V T 2= T
2)  T V F 2= T
3)  F V T 2= T
4)  F V F 2= F
```

This means a compound statement, which is the disjunction of two arguments, is false when and only when both arguments are false. It should be noted that this definition is sometimes called the 'inclusive or'. In real life we use the concept 'or' in both the inclusive and the exclusive sense, and assume that it is clear what is meant. The connective conditional is defined by the following table:

```
> | T F
—
T | T F
F | T T
```

or by the following equations:

```
1)  T > T 2= T
2)  T > F 2= F
3)  F > T 2= T
4)  F > F 2= T
```

This means that a compound statement, which is a conditional, is false when and only when the left argument is true and the right argument is false. In a conditional statement the left argument is often called the antecedent and the right argument, the consequent.This is difficult for most people to grasp, but in everyday speech, one normally does not use a false antecedent. Compound statement 10) above is an example of a conditional statement where the antecedent is false and the consequent is true, thus by the definition of the conditional connective, the statement 10): If 2+3=6, then 2+2=4, is an example of a true statement (F > T 2= T).

The connective biconditional can be defined by the following table:

```
= | T F
—
T | T F
F | F T
```

or by the following equations:

```
1)  T = T 2= T
2)  T = F 2= F
3)  F = T 2= F
4)  F = F 2= T
```

This means that a biconditional statement is true when and only when its arguments have the same truth value.

The unary connective negation can be defined by the equations:

```
~T 2= F
~F 2= T
```

A symbolization of 10) above is p > q where p represents: 2+3=6 and q represents: 2+2=4. A truth table analysis of the statement formula is:

```
         p q | p > q
         ———
case 1 T T | T*T*T
case 2 T F | T*F*F
case 3 F T | F*T*T
case 4 F F | F*T*F
```

The *'s are used to enclose the 'answer' column.

Now statement 10) would be a case 3 situation, hence is considered a true statement. Yes, we just said that the compound statement, If 2+3=6, then 2+2=4 , is a true statement.

Now I shall discuss the program. Statement letters are restricted to p,q,r, and s, and the connectives: negation, conjunction, disjunction, conditional, and biconditional are represented by the symbols: ~,&,V,>, and =, respectively. There are two parts to the truth tables generated. The first part is what I call the header and always consists of three lines. The second header line starts with an alphabetical listing of the statement letters used in the formula, followed by the formula itself.The first header line indicates the order the operations were performed in generating a truth table for the given formula, and the third header line is just a separator line consisting of numerous - characters. The second part of the truth table I call the body. The body consists of 2An lines, where n is the number of distinct statement letters used in the formula. The formulas allowed by this program must contain at least 1 statement letter, and can contain at most 4 distinct statement letters, therefore the body of a truth table generated by this program contains 2,4,8, or 16 lines. Let's look at the truth table generated by the program for the formula: p>(~q&p) . This formula contains 2 distinct statement letters, therefore its truth table body consists of 4 lines. The truth table is:

```
         0 3  1 0 2 0
    p q | (p > (~ q & p))
    ---------
    T T |  T*F* F T F T
    T F |  T*T* T F T T
    F T |  F*T* F T F F
    F F |  F*T* T F F F
```

The 3 lines below are the header. The header always consists of 3 lines.

```
h1:          0 3  1 0 2 0
h2:  p q | (p > (~ q & p))
h3:  ---------
```

The 4 lines below are the body of the truth table. The number of lines in the body is always a power of 2.

```
case 1:   T T |  T*F* F T F T
case 2:   T F |  T*T* T F T T
case 3:   F T |  F*T* F T F F
case 4:   F F |  F*T* T F F F
```

Case 1 represents the formula when both p and q have T as their value. Case 2 represents the formula when p has the value T and q has the value F, etc. .

The numbers in header line h1 represent the order the operations are performed. Zeros appear above the statement letters. In the above example the values in the body of the truth table were calculated as follows: first the value under ~ is calculated from the column under the q just to the right of ~, secondly the & value is calculated by using the value under the ~ and the column under the p just to the right of the & symbol, and thirdly the value under the > is calculated using the value under the p to the left and the value under the & to the right.

Two statement formulas are said to be logically equivalent if the biconditional between them is always true. The following two formulas are quite often used in programming: 1) pVq and 2) ~(~p&~q) and the biconditional between them produces a truth table with all true values in the 'answer' column, thus indicating that these two formulas are logically equivalent.

```
          0 1 0  6  5  2 0 4 3 0
    p q | ((p V q) = (~ (~ p & ~ q)))
    ------------------
    T T |   T T T *T* T  F T F F T
    T F |   T T F *T* T  F T F T F
    F T |   F T T *T* T  T F F F T
    F F |   F F F *T* F  T F T T F
```

The order that the operations are performed when no parentheses exist is: ~, &, V, >, and = . All ~'s are calculated before any &'s, all &'s are calculated before any V's etc. .The program scans left to right and as soon as an operation is calculated, the scan starts again from the left starting place. This process continues until no operations are left uncalculated. If a formula has parentheses, then the program finds the leftmost subformula containing no parentheses, evaluates it, removes the parentheses enclosing this subformula and then restarts. The program is finished when no parentheses exist. This works because the program encloses the formula that has been entered, in parentheses. A negation of a negation works a little differently because negation is a unary operator. Below is the truth table for (~~pV~q)=~~(~p>~q). See if you follow the order the operations were performed. The numbers 10, 11, 12,... , when necessary, are represented by A, B, C,... in header line 1. The biconditional (=) below is the tenth and last operation performed.

```
          2 1 0 4 3 0  A 9 8  5 0 7 6 0
    p q | ((~ ~ p V ~ q) = ~ ~ (~ p > ~ q))
    ---------------------
    T T |   T F T T F T *T*T F  F T T F T
    T F |   T F T T T F *T*T F  F T T T F
    F T |   F T F F F T *T*F T  T F F F T
    F F |   F T F T T F *T*T F  T F T T F
```

This article is by no means the last word on propositional logic. It, hopefully, will help some people and also the program will allow one to investigate the cases of a formula.

It should be noted that the if..., then... built into most computer languages is NOT a logical connective. The logical connectives built into the Microware C Language are !, &&, || and ==. They represent negation, conjunction, disjunction, and biconditional respectively.

P.S. In November 1988, I compiled this program using Turbo C with the indicated changes. Also since I'm going to Poland for four months, starting February 1989, I added the Polish notation routine. Polish notation is used to indicate an expression without the use of parentheses. Some calculators have reverse Polish notation capabilities.

I hope you have fun with the program.

+++

```
/* *****************************************************************

        Name:       Logic
        Date:       88/3/28
        Update;     88/11/7
        Author:     J. A. (Jim) Woodward
        Compiling:  logic.c
        Compiler:   Microware C Compiler
        ********************************

        Function: This program is an educational tool designed to
        help one learn  about the construction of truth tables.

  ******************************************************************/

/* for Turbo C do a global replace: index            */
/*                            with: strchr           */
/*                  global replace: putchar('\zc')   */
/*                            with: clrscr()         */

#include <stdio.h>
 /* IBMpc (with Turbo C Compiler) add: #include  <conio.h> */
#define  EOL  13
 /* IBMpc (with Turbo C Compiler) #define  EOL  10  */
char  oporder[80];  /* header line1 string  */
char  polish[50];
char  *index();
int  order,polk;          /* operation order indicator  */

main(argc)
int  argc;
{
char  expr[65],c,ch;  /*  expr[]:=logical expression string  */

escape();
putchar('\zc');  /*  Turbo C  clrscr()  */
if (argc>1)
  escape();
while (1) {
  c='s';
  while (-1==getexpr(expr));
  if(-1!=prteval(expr,c)) {;
    printf("\n\nWould you like to save a copy of the above?  <y , n> ");
    c='n';
    while ((ch=getchar())!=EOL)
      c=ch;
putchar('\zc'); /*  Turbo C  clrscr()  */
    prteval(expr,c);
  }
}
}

getexpr(s)
char s[];
{
int  len, i, j;

  for (i=1;i<65;i++)
      s[i]='\0';
s[0]='(';
while (1==strlen(s)) {
  printf("\n\nEnter the logic expression, please.\n");
  gets(&s[1]);
}
if (index(s,'?'))
   escape();
strcat(s,")");  /* complete enclosing logical string in parentheses*/
if (strcmp(s,"(zz)")==0)
  exit(0);
for (i=0,len=0;s[i]!='\0';i++)
   if (s[i]!=' ')
      s[len++]=s[i];
s[len]='\0';
if (3>len)
   return(-1);
for (i=1,j=0;i<len-1;i++) {  /*  see if parentheses are balanced  */
 if ('('==s[i])
  j++;
 if (')'==s[i])
  j-;
 if (j<0)
  return(error(s,1,i));
}
if (64<len)
  return(error (s,0,63));
if (0<j)
 return(error(s,1,i));
for (i=1;i<len-1;i++)
  switch  (s[i])   { /* syntax check on logical expression string  */
    case 'p':
    case 'q':
    case 'r':
    case 's':
      if (NULL!=*index("pqrs)",s[i-1]))
        return(error(s,3,i-1));
      if (NULL!=*index("pqrs~(",s[i+1]))
        return(error(s,3,i+1));
      break;
    case 'V':
    case '=':
    case '&':
    case '>':
      if (NULL!=*index("&=V>~(",s[i-1]))
        return(error(s,4,i-1));
      if (NULL!=*index("&=V>)",s[i+1]))
        return(error(s,4,i+1));
      break;
    case '~':
      if (NULL!=*index(")&=V>",s[i+1]))
        return(error(s,5,i+1));
      if (0>i && NULL!=*index(")pqrs",s[i-1]))
        return(error(s,5,i-1));
      break;
    case '(':
      if (NULL!=*index(")pqrs",s[i-1]))
        return(error(s,6,i-1));
      if (NULL!=*index(")&=V>",s[i+1]))
        return(error(s,6,i+1));
        break;
      case ')':
        if (NULL!=*index("(&=V>-~",s[i-1]))
          return(error(s,7,i-1));
        if (NULL!=*index("(pqrs~",s[i+1]))
          return(error(s,7,i+1));
        break;
      default:
        return(error(s,8,i));
    }
  return(0);  /* logical expression is OK  */
}

error(s,n,i)
char  *s;
int   n,i;
{
int  j;

putchar('\xc');  /*  Turbo C  clrscr()  */
printf("If you don't understand how this works enter a '?' when ");
printf("\nprompted for an expression.\n");
    s[0]=' ';
    s[strlen(s)-1]=' ';
    printf("\nERROR:");
    printf("\n%s\n",s);
    for (j=0;j<i;j++)
      putchar(' ');
    putchar('^');
  switch (n)  {

    case 0:
      printf("\nExpression is TOO long!");
      return(-1);
    case 1:
      printf("\nUnbalanced parentheses!");
      return(-1);
    case 2:
      printf("\nBad character!");
      return(-1);
    case 3:
      printf("\ntype:  pqrs");
      return(-1);
    case 4:
      printf("\ntype:  &=V>");
      return(-1);
    case 5:
      printf("\ntype:  ~");
      return(-1);
    case 6:
      printf("\ntype:  (");
      return(-1);
    case 7:
      printf("\ntype:  )");
      return(-1);
    case 8:
      printf("\ntype: Bad character");
      return(-1);
  }
}

prteval(s1,c)
char  *s1,c;
{
int   cases=1,pcase, vars=0,i,j,k,len, pos,pol;
char  var[5],string[81],valstr[81],*p,s[66],path[30];
FILE  *fp;

  if ('s'!=c && 'y'!=c)
      return(0);
  if ('y'==c) {
      printf("\nEnter path to file, please.  ");
      gets(path);
      fp=fopen(path,"a");
      putc('\n',fp);
  }
  else
      fp=stdout;
  putchar('\xc');  /*  Turbo C  clrscr()  */
  strcpy(var,"pqrs");
  strcpy(s,s1);
  for (i=0;i<79;i++)  /* initialize header line1  */
     oporder[i]=' ';
  oporder[79]='\0';
  order=1;    /* initialize order indicator  */
  k=0;
  for (j=0;j<4;j++)
    for (i=1;s[i]!='\0';i++)
       if (s[i]==var[j])  {
          var[k++]=var[j];
          break;
       }
  var[k]='\0';
  vars=strlen(var);  /* count number of distinct varibles used  */
  for (i=0;i<vars;i++) { /* calculate the number of cases needed  */
    cases*=2;
    string[2*i]=var[i]; /* making header line2  */
    string[2*i+1]=' ';
  }
  string[2*vars]='|';
  string[2*vars+1]=' ';
  j=2*vars+2;
  for (i=0;s[i]!='\0';i++)
    if (NULL!=*index("&=V>",s[i])) {
                          /* put spaces around binary operations */
      string[j++]=' ';
      string[j++]=s[i];
      string[j++]=' ';
    }
    else if ('~'==s[i]) {  /*  put space to right of ~  */
      string[j++]='~';
      string[j++]=' ';
    }
    else
      string[j++]=s[i];
  string[j]='\0'; /* string[] is now header line2  */
  len=j;
  if (79<len)
```

```
        return(error(string,0,79));
    for (i=0;i<cases;i++)  { /* making case strings (valstr[])  */
      for (k=0;k<79;k++)
         valstr[k]=' ';
      valstr[79]='\0';
      pcase=i;
      for (j=0;j<vars;j++)  { /* calculate values for variables  */
         pos=2*(vars-j-1);
         valstr[pos]=(0==pcase%2?'T':'F');
         for (k=2*vars;k<79 && string[k]!='\0';k++)
            if (string[k]==string[pos])
              valstr[k]=valstr[pos];
            else if (NULL==*index("pqrs ",string[k]))
               valstr[k]=string[k];
         pcase/=2;
         valstr[2*(vars-j)-1]=' ';
      }
      while ((p=index(valstr,')'))!=NULL) {
         j=p-valstr; /* locate first right parenthesis and its mate  */
         for (pos=j-1;valstr[pos]!='(';pos-)
              ;
         while (-1==bicop(valstr,pos,j,i));
                              /* calculate the case string */
         valstr[pos]=valstr[j]=' ';
                            /* replace parentheses with spaces  */
      }
      for (k=pos;k<j;k++)
         if (valstr[k]=='T' || valstr[k]=='F') {
            pol=k;
            valstr[k-1]=valstr[k+1]='*'; /* place *'s around answer */
            break;
         }
      for (k=pos;k<len;k++)
         if (valstr[k]=='t' || valstr[k]=='f')
            valstr[k]=(valstr[k]=='t'?'T':'F'); /* recapitalize */
      valstr[len]='\0';
      if (0==i) {
         for (k=2*vars+3;string[k]!='\0';k++)
           if (NULL!=*index(var,string[k]))
              oporder[k]='0';
         oporder[len]='\0';
         fprintf(fp,"\n%s",oporder); /* print header  */
         fprintf(fp,"\n%s\n",string);
         for (k=0;k<len;k++)
           putc('-',fp);
      }
      fprintf(fp,"\n%s",valstr); /* print case  */
    }
   polk=0;
   polnotn(string,pos,len-1,pol);
   fprintf(fp,"\n\n%s in Polish notation is:\n %s",s,polish);
   if (fp!=stdout) {
     fprintf(fp,"\n");
     fclose(fp);
   }
   return(0);
}

not(s,pos,len,kase)
char  *s;
int   pos,len,kase;
{
int i=0,j;

  for (i=pos;s[i]!='-' && i<len;i++)
     ;
  if (i==len)
       return(0);
  for (j=i+1;j<len;j++) {
      if ( '~'==s[j])
         i=j;
      else if (s[j]=='T' || s[j]=='F') {
        s[i]=(s[j]=='T'?'F':'T');
        s[j]=tolower(s[j]);     /* make used value lower k */
        if (0==kase) {
          if (order<10)   /* used for operation order  value  */
             oporder[i]='0'+order++;
          else
             oporder[i]='A'-10+order++;
        }
        return(-1);
      }
  }
return(0);
}

bicop(s,pos,len,kase)
char  *s;
int pos,len,kase;
{
int  h,i,j,k;
static char op[]="&V>=";

  while (-1==not(s,pos,len,kase));
  for (h=0;h<4;h++) {
    for (j=pos;s[j]!=op[h] && j<len;j++)
       ;
    if (j==len )
      if (3==h)
        return(0);
      else
         continue;
    k=j+1;
    while (s[k]!='T' && s[k]!='F')
      k++;
    i=j-1;
    while (s[i]!='T' && s[i]!='F')
       i-;
  if (3==h){
    s[j]=(s[i]==s[k]?'T':'F');
  }
  else if (2==h) {
    s[j]=(s[i]=='T' && s[k]=='F'?'F':'T');
  }
  else if (1==h) {
    s[j]=(s[i]=='T' || s[k]=='T'?'T':'F');
  }
  else if (0==h) {
      s[j]=(s[i]=='F' || s[k]=='F'?'F':'T');
    }
      s[i]=tolower(s[i]);
      s[k]=tolower(s[k]);
      if (0==kase)  {
        if (order<10)
           oporder[j]='0'+order++;
        else
           oporder[j]='A'-10+order++;
      }
      return(-1);
    }
}

polnotn(st,lft,rght,pol)
char  st[];
int   lft,rght,pol;
{
char  lp,rp;
int   i,j;


pollsh[polk++]=st[pol];
if (1==polk && '0'==oporder[pol]) {
   pollsh[polk]='\0';
   return(0);
}
lp=rp='0';
if (st[pol]=='~') {
   for (i=pol+1,j=1;i<rght ;i++)
           /* find the largest oporder to the right */
      if (rp<=oporder[i]) {
        rp=oporder[i];
        j=i;
      }
   if ( rp!='0')
     polnotn(st,pol+1,rght,j);
   else {
     pollsh[polk++]=st[j];
     pollsh[polk]='\0';
   }
}
else {
  for (i=lft,j=1;i<pol;i++] /* find the largest oporder to the left */
     if (oporder[i]>=lp) {
       lp=oporder[i];
       j=i;
     }
  if ('0'!=lp)
     polnotn(st,lft,pol-1,j);
  else
     polish[polk++]=st[j];
  for (i=pol+1,j=i;i<rght;i++)
                       /* find the largest oporder to the right */
    if (oporder[i]>=rp)  {
      rp=oporder[i];
      j=i;
    }
  if ('0'!=rp)
      polnotn(st,pol+1,rght,j);
    else {
      polish[polk++]=st[j];
      polish[polk]='\0';
    }
}
}
escape()
{
printf("\nUsage: This program generates truth tables.\n");
printf("       You may use only the letters: 'p','q','r',or's'");
printf(" as statement variables.");
printf("\n        The connectives are symbolized as follows:");
printf("\n           Biconditional is '=' (if and only if)");
printf("\n           Conditional is '>' (if...,then...)");
printf("\n           Conjunction is '&' (and)");
printf("\n           Disjunction is 'V' (or)");
printf("\n           Negation is '~' (not)");
printf("\nExample1: p>(qVr) would be read \"If p, then q or r\"");
printf("\nExample2: p&(qVr) would be read \"p, and q or r\""];
printf("\nExample3: (p&q)Vr would be read \"p and q, or r\"");
printf("\nExample4: p=~q would be read \"p if and only if not q\"");
printf("\nTo leave the program, enter the string 'zz'  when prompted");
printf(" for an expression.");
printf("\nThis Program was written by: Jim Woodward, Lock Haven, PA 17745");
printf("\nPRESS ENTER to continue."];
getchar();
}

+++


echo 'logic.c'
-x
echo c.prep:
C.PREP logic.c >ctmp.4.m
x
echo c.pass1:
C.PASS1 ctmp.4.m -o=ctmp.4.i
del ctmp.4.m
echo c.pass2:
C.PASS2 ctmp.4.i -o=ctmp.4.a
del ctmp.4.i
echo c.opt:
C.OPT ctmp.4.a ctmp.4.o
del ctmp.4.a
echo   rma:
  RMA ctmp.4.o -o=ctmp.4.r
del ctmp.4.o
echo  rlink:
 RLINK /r0/lib/cstart.r ctmp.4.r -o=logic.bin -l=/r0/lib/clib.l
del ctmp.4.r

+++
```

FOR THOSE WHO NEED TO KNOW

68 MICRO JOURNAL™

# Bit-Bucket

**By: All of us**

*"Contribute Nothing - Expect Nothing"*, DMW '86

**MOTOROLA INC.**

Microprocessor Products Group
6501 William Cannon Drive West
Austin, Texas 78735-8598

Zachary Nelson
Cunningham Communication, Inc.
(408) 982-0400

Dean Mosley
Microprocessor Products Group
(512) 440-3095

### Arix Introduces High Performance 68020-Based System

Multiprocessor System Delivers 30 MIPS Performance
System Uses Up to 84 Motorola Processors for Central, Peripheral Functions

SAN JOSE, Calif., Jan. 24, 1989 — Arix Corp. today introduced a series of high-performance business computers based on multiple Motorola 68000 family processors. Arix's new System90 series uses up to eight 25 MHz 68020 (020) processors to deliver 30 MIPS (million instructions per second) of performance. In total, the computer uses up to 84 68000 family chips to control central processing, input/output (I/O) and disk control functions.

System90 configurations use the Unix® operating system and can support from 32 to 512 users. Arix's new top-of-the-line Model 80 uses eight 020s for central processing functions, sixty 020s for I/O control and sixteen 68000s for disk control. The combined processing power of these eighty-four Motorola chips is over 200 MIPS.

"The new Arix systems show the versatility and price-performance of our 68000 family of processors," said Murray A. Goldman, senior vice president and general manager of Motorola's Microprocessor Products Group (Austin, Texas). "The flexibility of the 68000 line has made it a dominant engine in both the embedded control and central processing markets."

The 68000 line currently has four microprocessor members — the 68000, 68010, 68020 and 68030. The current development of the next generation (68040) continues the evolution of the processor family. In total, 21 million chips from the 68000 family have been sold in applications including supercomputers, high-end workstations, business computers and embedded control devices. All generations of the 68000 are compatible with each other. Software written for one chip runs with no modification on the others, and hardware upgrades are very simple.

Arix Corp. (San Jose, Calif.) designs, manufactures and markets high-performance, Unix-based multiprocessor computer systems. Arix distributes its systems almost exclusively through domestic and international reseller channels, including original equipment manufacturers, value-added resellers and systems integrators. Arix common stock is traded on the NASDAQ National Market Systems under the symbol ARIX.

### MOTOROLA OFFERS USER'S MANUALS FOR 88000 RISC PROCESSORS

AUSTIN, Texas, Jan. 18, 1989 — Motorola's Microprocessor Products Group today announced the availability of two user's manuals for its 88000 RISC (reduced instruction set computer) microprocessor family. This announcement closely follows last week's news of Motorola's 88000 being available for general sampling.

The new user's manuals provide detailed technical information on Motorola's 88100 central processing unit (CPU) and 88200 cache/memory management unit (CMMU). System designers and software developers will reference the manuals for explanations of the addressing modes and instruction sets, analyses of bus operations and register usage, data on electrical characteristics and outlines of minimum system configurations.

### MOTOROLA BEGINS GENERAL SAMPLING OF 88000 RISC MICROPROCESSOR FAMILY

Volume Production to Begin in Second Quarter

AUSTIN, Texas, Jan. 10, 1989 — Motorola's Microprocessor Products Group today announced that its new family of high-performance RISC (reduced instruction set computer) microprocessors, the 88000, is available for general sampling. Motorola also reports that the product will enter full production in the second quarter of 1989.

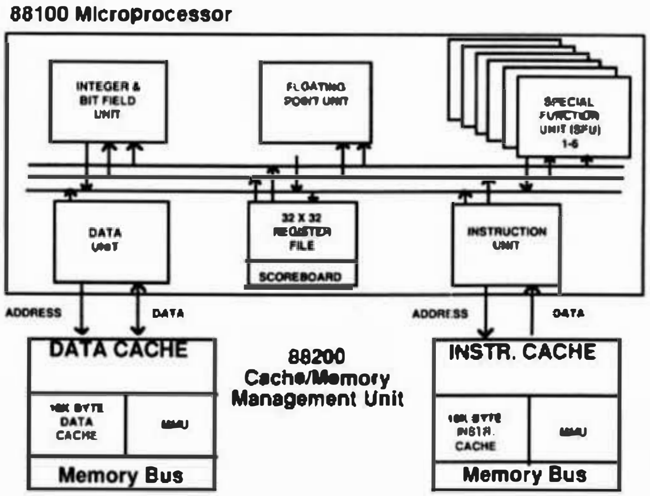
Today's announcement of general sampling brings the very high performance of the 88000 microprocessor architecture to the general marketplace. More than fifty customers have received product samples since early 1988. These customers worked closely with Motorola to evaluate and maximize the processor's functionality and performance. Motorola is now accepting orders for general delivery.

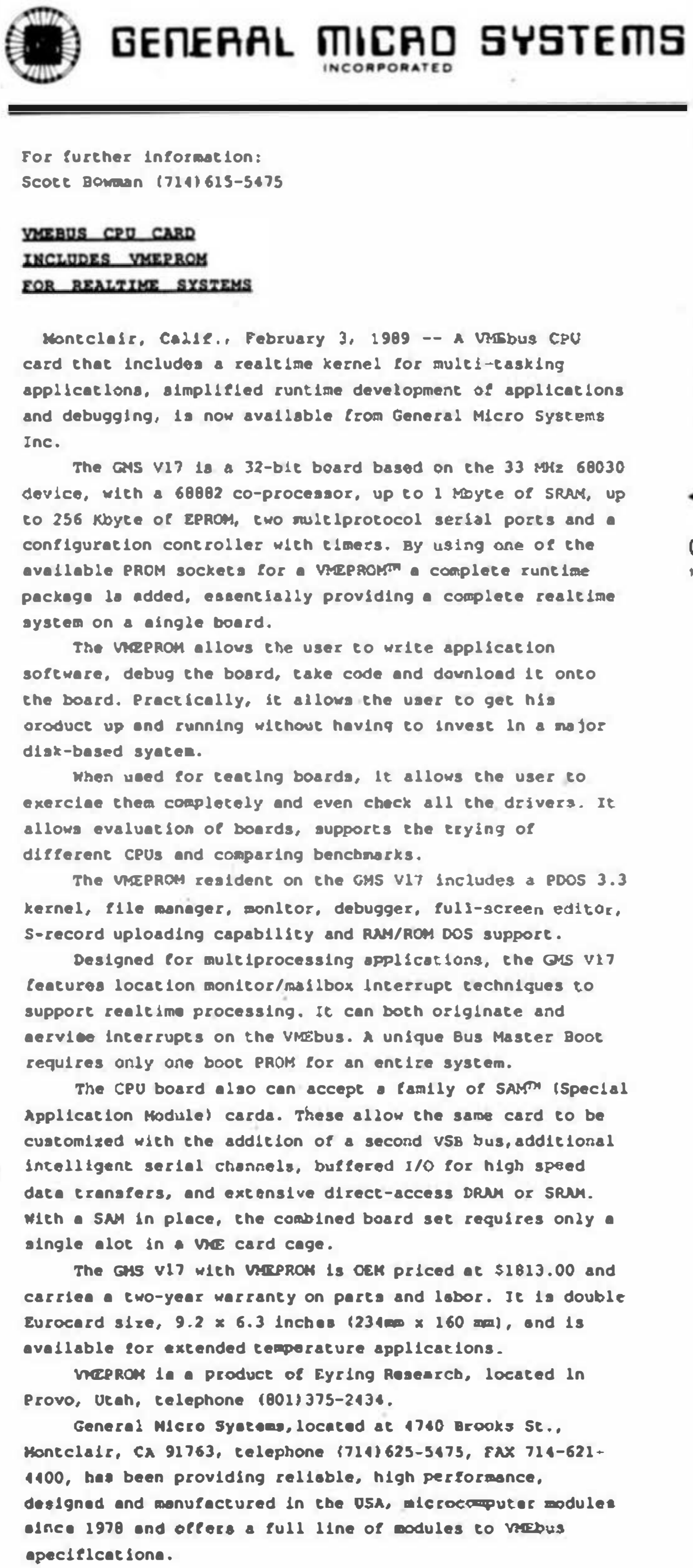
"The 88000 is designed to integrate the features required by computer makers on silicon to reduce system design time and cost while markedly increasing performance," said Murray A. Goldman, senior vice president and general manager of Motorola's Microprocessor Products Group. "Our manufacturing strength has allowed us to create the most complete and aggressive microprocessor implementation ever offered to computer manufacturers."

The 88000 family includes two chips. The 88100 central processing unit integrates integer and floating point capabilities. The 88200 Cache/Memory Management Unit (CMMU) provides a complete, single-chip solution to cache memory design, an essential function in RISC computer systems. Together the two chips provide functions equivalent to 50 or more components in competing systems.

Sample quantities of the 88100 central processing unit are available for $494. The 88200 is available at $619. Inquiries should be directed to local Motorola sales offices. Full production will begin in the second quarter.

**88000 BLOCK DIAGRAM**





GENERAL MICRO SYSTEMS
INCORPORATED

For further information:
Scott Bowman (714)615-5475

**VMEBUS CPU CARD**
**INCLUDES VMEPROM**
**FOR REALTIME SYSTEMS**

Montclair, Calif., February 3, 1989 -- A VMEbus CPU card that includes a realtime kernel for multi-tasking applications, simplified runtime development of applications and debugging, is now available from General Micro Systems Inc.

The GMS V17 is a 32-bit board based on the 33 MHz 68030 device, with a 68882 co-processor, up to 1 Mbyte of SRAM, up to 256 Kbyte of EPROM, two multiprotocol serial ports and a configuration controller with timers. By using one of the available PROM sockets for a VMEPROM™ a complete runtime package is added, essentially providing a complete realtime system on a single board.

The VMEPROM allows the user to write application software, debug the board, take code and download it onto the board. Practically, it allows the user to get his oroduct up and running without having to invest in a major disk-based system.

When used for testing boards, it allows the user to exercise them completely and even check all the drivers. It allows evaluation of boards, supports the trying of different CPUs and comparing benchmarks.

The VMEPROM resident on the GMS V17 includes a PDOS 3.3 kernel, file manager, monitor, debugger, full-screen editor, S-record uploading capability and RAM/ROM DOS support.

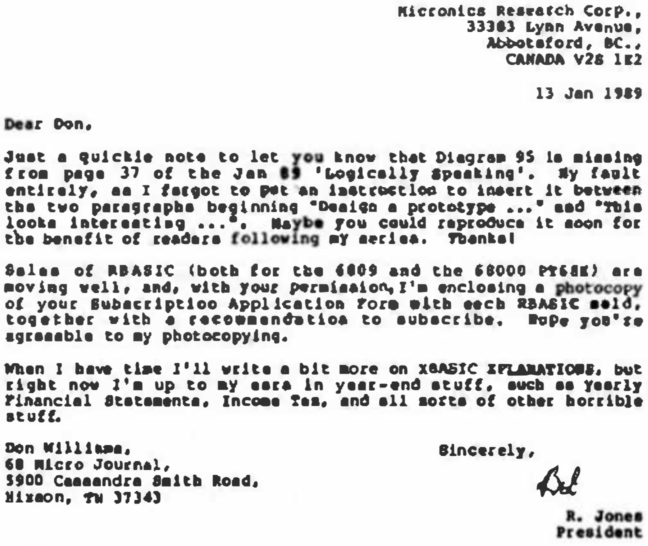
Designed for multiprocessing applications, the GMS V17 features location monitor/mailbox interrupt techniques to support realtime processing. It can both originate and aervice interrupts on the VMEbus. A unique Bus Master Boot requires only one boot PROM for an entire system.

The CPU board also can accept a family of SAM™ (Special Application Module) cards. These allow the same card to be customized with the addition of a second VSB bus, additional intelligent serial channels, buffered I/O for high speed data transfers, and extensive direct-access DRAM or SRAM. With a SAM in place, the combined board set requires only a single slot in a VME card cage.

The GMS V17 with VMEPROM is OEM priced at $1813.00 and carries a two-year warranty on parts and labor. It is double Eurocard size, 9.2 x 6.3 inches (234mm x 160 mm), and is available for extended temperature applications.

VMEPROM is a product of Eyring Research, located in Provo, Utah, telephone (801)375-2434.

General Micro Systems, located at 4740 Brooks St., Montclair, CA 91763, telephone (714)625-5475, FAX 714-621-4400, has been providing reliable, high performance, designed and manufactured in the USA, microcomputer modules since 1978 and offers a full line of modules to VMEbus specifications.



Micronics Research Corp.,
33383 Lynn Avenue,
Abbotsford, BC.,
CANADA V2S 1K2

13 Jan 1989

Dear Don,

Just a quickie note to let you know that Diagram 95 is missing from page 37 of the Jan 89 'Logically Speaking'. My fault entirely, as I forgot to put an instruction to insert it between the two paragraphs beginning "Design a prototype ..." and "This looks interesting ...". Maybe you could reproduce it soon for the benefit of readers following my series. Thanks!

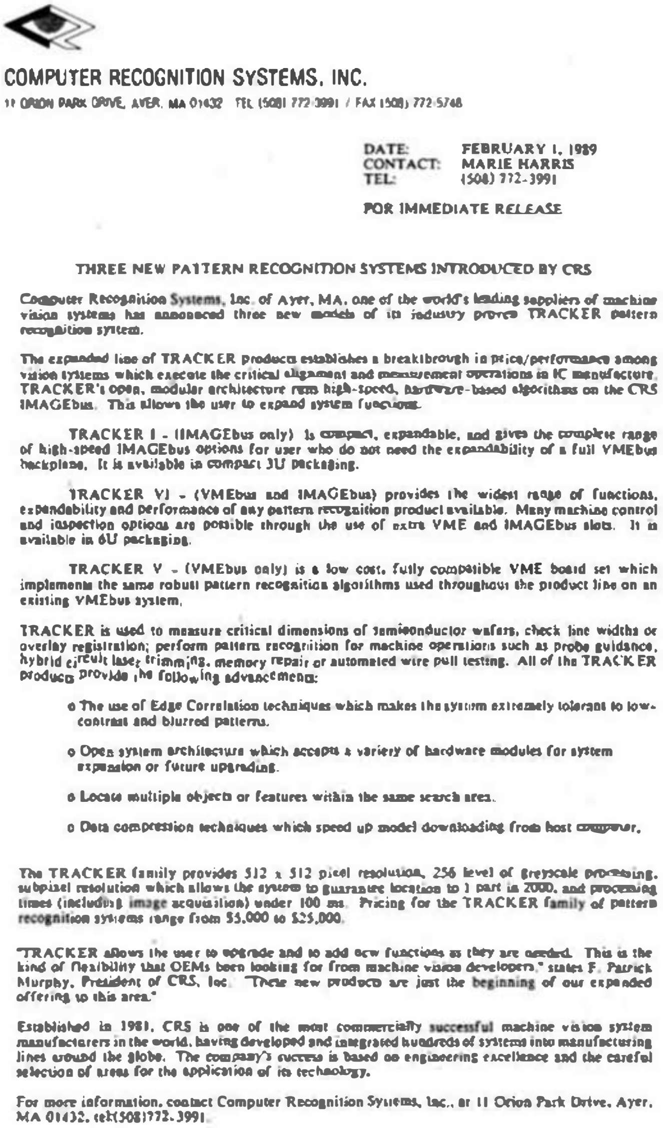
Sales of RBASIC (both for the 6809 and the 68000 PT68K) are moving well, and, with your permission, I'm enclosing a photocopy of your Subscription Application Form with each RBASIC sold, together with a recommendation to subscribe. Hope you're agreeable to my photocopying.

When I have time I'll write a bit more on XBASIC EXPLANATIONS, but right now I'm up to my ears in year-end stuff, such as Yearly Financial Statements, Income Tax, and all sorts of other horrible stuff.

Don Williams,
68 Micro Journal,
5900 Cassandra Smith Road,
Hixson, TN 37343

Sincerely,

R. Jones
President



COMPUTER RECOGNITION SYSTEMS, INC.
11 ORION PARK DRIVE, AYER, MA 01432   TEL (508) 772-3991 / FAX (508) 772-5748

DATE: FEBRUARY 1, 1989
CONTACT: MARIE HARRIS
TEL: (508) 772-3991

FOR IMMEDIATE RELEASE

THREE NEW PATTERN RECOGNITION SYSTEMS INTRODUCED BY CRS

Computer Recognition Systems, Inc. of Ayer, MA, one of the world's leading suppliers of machine vision systems has announced three new models of its industry proven TRACKER pattern recognition system.

The expanded line of TRACKER products establishes a breakthrough in price/performance among vision systems which execute the critical alignment and measurement operations in IC manufacture. TRACKER's open, modular architecture runs high-speed, hardware-based algorithms on the CRS IMAGEbus. This allows the user to expand system functions.

TRACKER I - (IMAGEbus only) is compact, expandable, and gives the complete range of high-speed IMAGEbus options for user who do not need the expandability of a full VMEbus backplane. It is available in compact 3U packaging.

TRACKER VI - (VMEbus and IMAGEbus) provides the widest range of functions, expandability and performance of any pattern recognition product available. Many machine control and inspection options are possible through the use of extra VME and IMAGEbus slots. It is available in 6U packaging.

TRACKER V - (VMEbus only) is a low cost, fully compatible VME board set which implements the same robust pattern recognition algorithms used throughout the product line on an existing VMEbus system.

TRACKER is used to measure critical dimensions of semiconductor wafers, check line widths or overlay registration; perform pattern recognition for machine operations such as probe guidance, hybrid circuit laser trimming, memory repair or automated wire pull testing. All of the TRACKER products provide the following advancements:

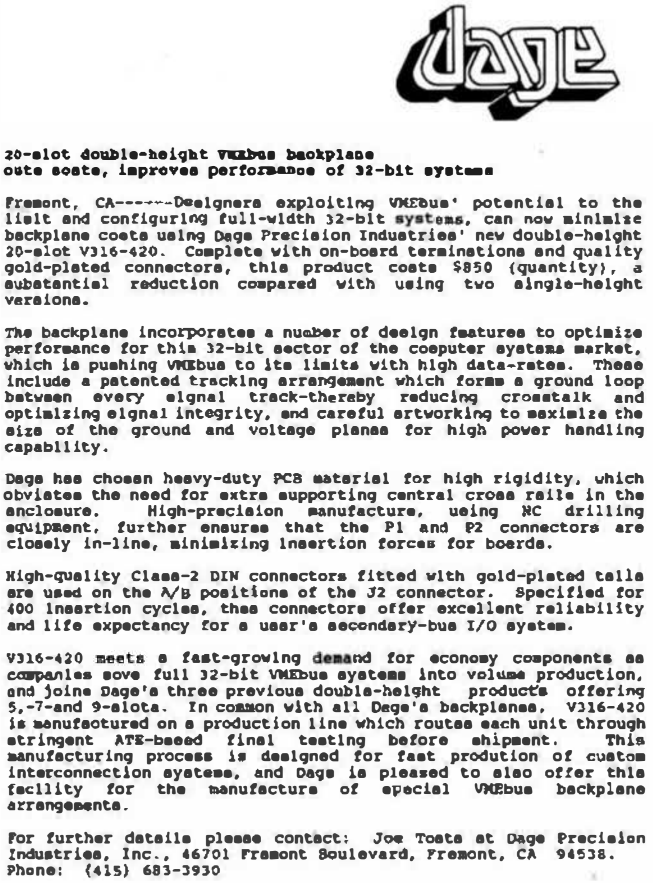
- The use of Edge Correlation techniques which makes the system extremely tolerant to low-contrast and blurred patterns.
- Open system architecture which accepts a variety of hardware modules for system expansion or future upgrading.
- Locate multiple objects or features within the same search area.
- Data compression techniques which speed up model downloading from host computer.

The TRACKER family provides 512 x 512 pixel resolution, 256 level of greyscale processing, subpixel resolution which allows the system to guarantee location to 1 part in 2000, and processing times (including image acquisition) under 100 ms. Pricing for the TRACKER family of pattern recognition systems range from $5,000 to $25,000.

"TRACKER allows the user to upgrade and to add new functions as they are needed. This is the kind of flexibility that OEMs been looking for from machine vision developers," states F. Patrick Murphy, President of CRS, Inc. "These new products are just the beginning of our expanded offering to this area."

Established in 1981, CRS is one of the most commercially successful machine vision system manufacturers in the world, having developed and integrated hundreds of systems into manufacturing lines around the globe. The company's success is based on engineering excellence and the careful selection of areas for the application of its technology.

For more information, contact Computer Recognition Systems, Inc., at 11 Orion Park Drive, Ayer, MA 01432, tel(508)772-3991.



dage

**20-slot double-height VMEbus backplane cuts costs, improves performance of 32-bit systems**

Fremont, CA------Designers exploiting VMEbus' potential to the limit and configuring full-width 32-bit systems, can now minimize backplane costs using Dage Precision Industries' new double-height 20-slot V316-420. Complete with on-board terminations and quality gold-plated connectors, this product costs $850 (quantity), a substantial reduction compared with using two single-height versions.

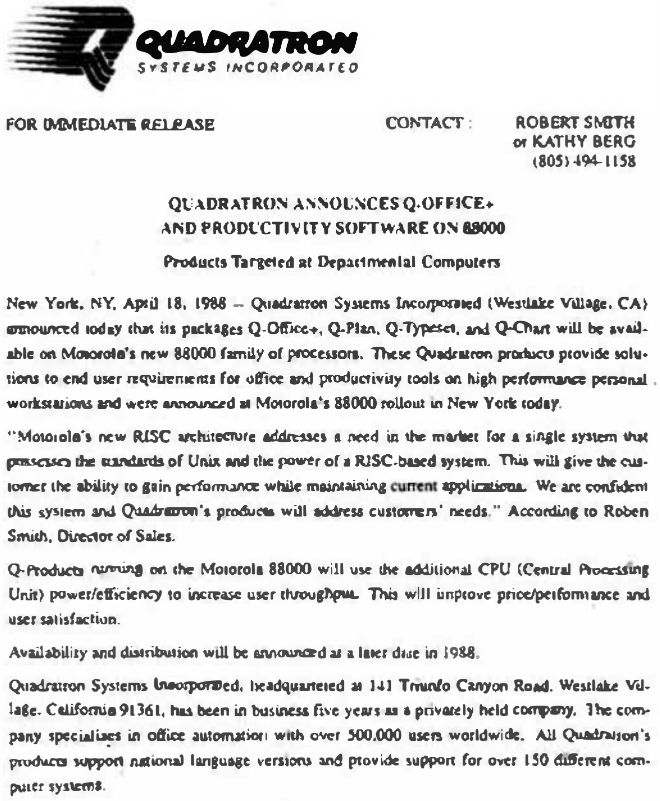
The backplane incorporates a number of design features to optimize performance for this 32-bit sector of the computer systems market, which is pushing VMEbus to its limits with high data-rates. These include a patented tracking arrangement which forms a ground loop between every signal track-thereby reducing crosstalk and optimizing signal integrity, and careful artworking to maximize the size of the ground and voltage planes for high power handling capability.

Dage has chosen heavy-duty PCB material for high rigidity, which obviates the need for extra supporting central cross rails in the enclosure. High-precision manufacture, using NC drilling equipment, further ensures that the P1 and P2 connectors are closely in-line, minimizing insertion forces for boards.

High-quality Class-2 DIN connectors fitted with gold-plated tails are used on the A/B positions of the J2 connector. Specified for 400 insertion cycles, thes connectors offer excellent reliability and life expectancy for a user's secondary-bus I/O system.

V316-420 meets a fast-growing demand for economy components as companies move full 32-bit VMEbus systems into volume production, and joins Dage's three previous double-height products offering 5,-7-and 9-slots. In common with all Dage's backplanes, V316-420 is manufactured on a production line which routes each unit through stringent ATE-based final testing before shipment. This manufacturing process is designed for fast prodution of custom interconnection systems, and Dage is pleased to also offer this facility for the manufacture of special VMEbus backplane arrangements.

For further details please contact: Joe Toste at Dage Precision Industries, Inc., 46701 Fremont Boulevard, Fremont, CA 94538. Phone: (415) 683-3930



QUADRATRON
SYSTEMS INCORPORATED

FOR IMMEDIATE RELEASE

CONTACT: ROBERT SMITH
or KATHY BERG
(805) 494-1158

**QUADRATRON ANNOUNCES Q-OFFICE+ AND PRODUCTIVITY SOFTWARE ON 88000**

**Products Targeted at Departmental Computers**

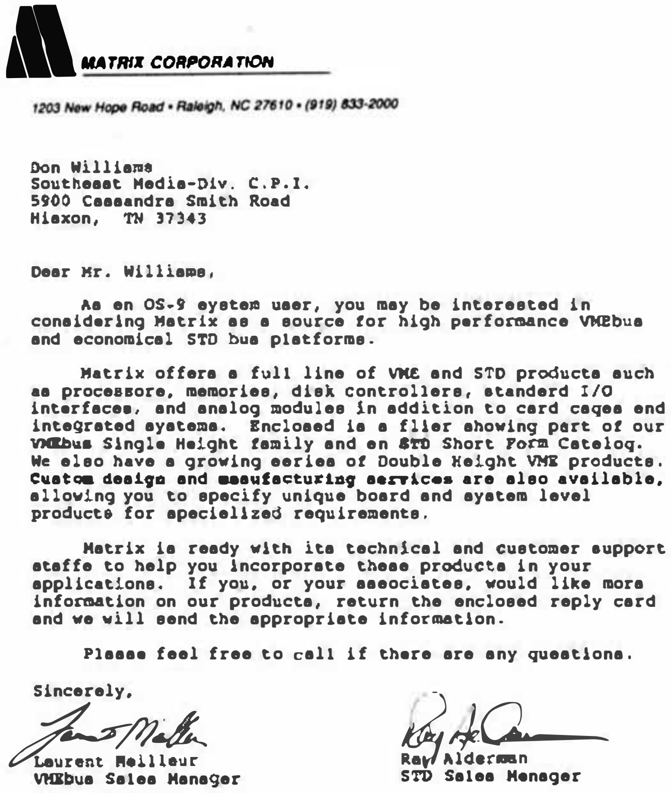
New York, NY, April 18, 1988 -- Quadratron Systems Incorporated (Westlake Village, CA) announced today that its packages Q-Office+, Q-Plan, Q-Typeset, and Q-Chart will be available on Motorola's new 88000 family of processors. These Quadratron products provide solutions to end user requirements for office and productivity tools on high performance personal workstations and were announced at Motorola's 88000 rollout in New York today.

"Motorola's new RISC architecture addresses a need in the market for a single system that possesses the standards of Unix and the power of a RISC-based system. This will give the customer the ability to gain performance while maintaining current applications. We are confident this system and Quadratron's products will address customers' needs." According to Robert Smith, Director of Sales.

Q-Products running on the Motorola 88000 will use the additional CPU (Central Processing Unit) power/efficiency to increase user throughput. This will improve price/performance and user satisfaction.

Availability and distribution will be announced at a later date in 1988.

Quadratron Systems Incorporated, headquartered at 141 Triunfo Canyon Road, Westlake Village, California 91361, has been in business five years as a privately held company. The company specializes in office automation with over 500,000 users worldwide. All Quadratron's products support national language versions and provide support for over 150 different computer systems.



MATRIX CORPORATION

1203 New Hope Road • Raleigh, NC 27610 • (919) 833-2000

Don Williams
Southeast Media-Div. C.P.I.
5900 Cassandra Smith Road
Hixson, TN 37343

Dear Mr. Williams,

As an OS-9 system user, you may be interested in considering Matrix as a source for high performance VMEbus and economical STD bus platforms.

Matrix offers a full line of VME and STD products such as processors, memories, disk controllers, standard I/O interfaces, and analog modules in addition to card cages and integrated systems. Enclosed is a flier showing part of our VMEbus Single Height family and an STD Short Form Catalog. We also have a growing series of Double Height VME products. Custom design and manufacturing services are also available, allowing you to specify unique board and system level products for specialized requirements.

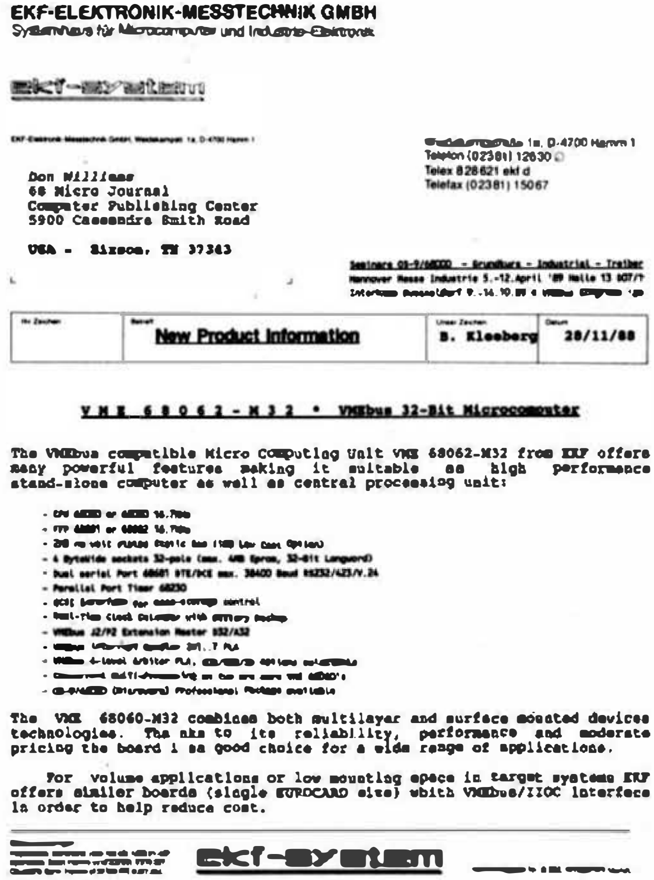
Matrix is ready with its technical and customer support staffs to help you incorporate these products in your applications. If you, or your associates, would like more information on our products, return the enclosed reply card and we will send the appropriate information.

Please feel free to call if there are any questions.

Sincerely,

Laurent Meilleur
VMEbus Sales Manager

Ray Alderman
STD Sales Manager



EKF-ELEKTRONIK-MESSTECHNIK GMBH
Systemhaus für Microcomputer und Industrie-Elektronik

ekf-system

EKF-Elektronik-Messtechnik GmbH, Wiedenkampstr. 1a, D-4700 Hamm 1

Don Williams
68 Micro Journal
Computer Publishing Center
5900 Cassandra Smith Road

USA - Hixson, TN 37343

Wiedenkampstraße 1a, D-4700 Hamm 1
Telefon (02381) 12030
Telex 828621 ekf d
Telefax (02381) 15067

Seminars OS-9/68000 - Grundkurs - Industrial - Treiber
Hannover Messe Industrie 5.-12.April '89 Halle 13 [illegible]
Interkama Duesseldorf 9.-14.10.89 + VMEbus Congress '89

| Ihr Zeichen | Betreff | Unser Zeichen | Datum |
|---|---|---|---|
|  | New Product Information | B. Kleeberg | 28/11/88 |

**VME 68062-M32 • VMEbus 32-Bit Microcomputer**

The VMEbus compatible Micro Computing Unit VME 68062-M32 from EKF offers many powerful features making it suitable as high performance stand-alone computer as well as central processing unit:

- CPU 68020 or 68030 16.7MHz
- FPP 68881 or 68882 16.7MHz
- 2/8 no wait state Static Ram (1MB Low Cost Option)
- 4 Bytewide sockets 32-pole (max. 4MB Eprom, 32-Bit Longword)
- Dual serial Port 68681 DTE/DCE max. 38400 Baud RS232/423/V.24
- Parallel Port Timer 68230
- SCSI Interface for mass-storage control
- Real-Time Clock Calendar with battery backup
- VMEbus J2/P2 Extension Master D32/A32
- VMEbus Interrupt Handler IR1..7 [illegible]
- VMEbus 4-Level Arbiter FLA, [illegible] options selectable
- [illegible] multi-processing [illegible] 68020's
- OS-9/68020 (Microware) Professional Package available

The VME 68060-M32 combines both multilayer and surface mounted devices technologies. Thanks to its reliability, performance and moderate pricing the board i sa good choice for a wide range of applications.

For volume applications or low mounting space in target systems EKF offers similar boards (single EUROCARD size) which VMEbus/IIOC interface in order to help reduce cost.

ekf-system

UNIFY
CORPORATION

Anu Shukla
Unify Corporation
(916) 920-9092

Linda Hayes
Motorola Computer Systems
(408) 864-4480

## MOTOROLA AND UNIFY ANNOUNCE JOINT SALES/MARKETING PACT

Cupertino, CA January 11, 1989 -- Motorola Inc., Computer Systems Division, and Unify Corporation jointly announced today a sales and marketing agreement under which the companies will cooperate in a worldwide effort to offer Unify's products on Motorola's System 8000 multiuser, super microcomputers. The agreement also applies to Motorola's future RISC-based systems.

Specifically the agreement provides for Motorola to receive early access to Unify's recently announced ACCELL/SQL and UNIFY 2000 products in addition to foreign language translations of ACCELL and the UNIFY RDBMS. The agreement is structured to promote cooperation between the two sales forces.

Commenting on the benefits of the agreement, David Saykally, Unify president and CEO, said, "The power and flexibility of ACCELL and UNIFY coupled with the price performance of the Motorola System 8000 make an unbeatable combination. We are delighted at forming this strategic relationship with Motorola."

Wayne Sennett, vice president and general manager of Motorola's Computer Systems Division, stated, "In keeping with our commitment to total customer satisfaction, Motorola is very pleased to offer these tools.

UNIFY 2000 gives Motorola customers a high performance, 100 percent uptime, DBMS for their large UNIX applications, while ACCELL/SQL provides a standard development technology for UNIX applications.

ACCELL consists of a tightly integrated application generator and a 4GL designed to interface with popular SQL relational databases. UNIFY is a high performance RDBMS designed for OLTP database applications requiring 100 percent uptime. It features five different data access methods, has multiple configuration options, and provides on-line backup and recovery as well as five-level security.

Motorola's System 8000 is a UNIX-based family of multiuser computers for workgroups. The family models range from low-end, six-user members to high-end, 128+ user systems.

Unify Corporation is a leading manufacturer of UNIX based application development software tools. Based in Sacramento, CA, the company employs 150 people in 15 offices across the United States, Europe, and Japan.

Motorola Computer Systems Division is based in Cupertino, CA, and develops and markets mid-range, multiuser super microcomputer systems for end users and VARs. With sales offices throughout the country and customers that include over half of the Fortune 1000, the division's focus is to provide total hardware and software systems solutions for the workgroup.

• • • •

Motorola and the Motorola logo are registered trademarks of Motorola, Inc.

UNIX is registered trademark of AT&T.

UNIFY and ACCELL are trademarks of Unify Corporation.

FORCE
FORCE COMPUTERS INC
3165 Winchester Blvd.
Campbell, CA 95008

**Prepared By:**
Shohat & Kahn PR
2959 S. Winchester Blvd, Campbell, CA 95008
Murry Shohat (408) 379-7434

**Contacts:**
FORCE USA: Wayne Fischer (408) 370-6300
FORCE GmbH: Anton Nausch (089) 600-910

## New Single-Board VMEbus Computer Molds Cost to Performance; SCSI Is Standard, Ethernet® is Optional

**CAMPBELL, CA., January 24, 1989** — A new 32-bit 68020-based single board computer will serve a very wide variety of industrial applications where users can select just the right combination of processing power and input/output capability. The CPU-27 can be ordered to run at 12.5, 16.7 or 25MHz. Its I/O capability includes SCSI in addition to many serial and parallel lines; Ethernet control is optional.

The CPU-27 is suited for both real-time and standard applications under a variety of operating systems. The board comes equipped with its own real-time kernel plus monitor/debugger for immediate operation.

Wayne Fischer, Director of Marketing, said "the industrial control and factory automation markets have been asking for a mid-range single-board computer that can slide up the cost-to-performance curve in terms of speed and functions. The CPU-27 can be purchased as either a modest 32-bit performer, as a computer with a very high level of functionality or at several points in between." Fischer indicated that industrial users seeking their first VMEbus solution would find the CPU-27 "an excellent growth vehicle."

The CPU-27 employs a single chip solution to VMEbus interface and control. "Our advanced VME/PLUS™ architecture merges high-density application-specific gate array technology with precision surface-mount board manufacturing to offer more functions in a single board computer for less dollars," said Fischer.

### Wide Applications Range from Standard & Optional Features

As a single board computer, the CPU-27 permits a wide range of I/O on its SCSI interface. Devices that can be accommodated include hard and floppy drives, tape units and optical disks, among others. A long period of compatible growth is envisioned for SCSI devices.

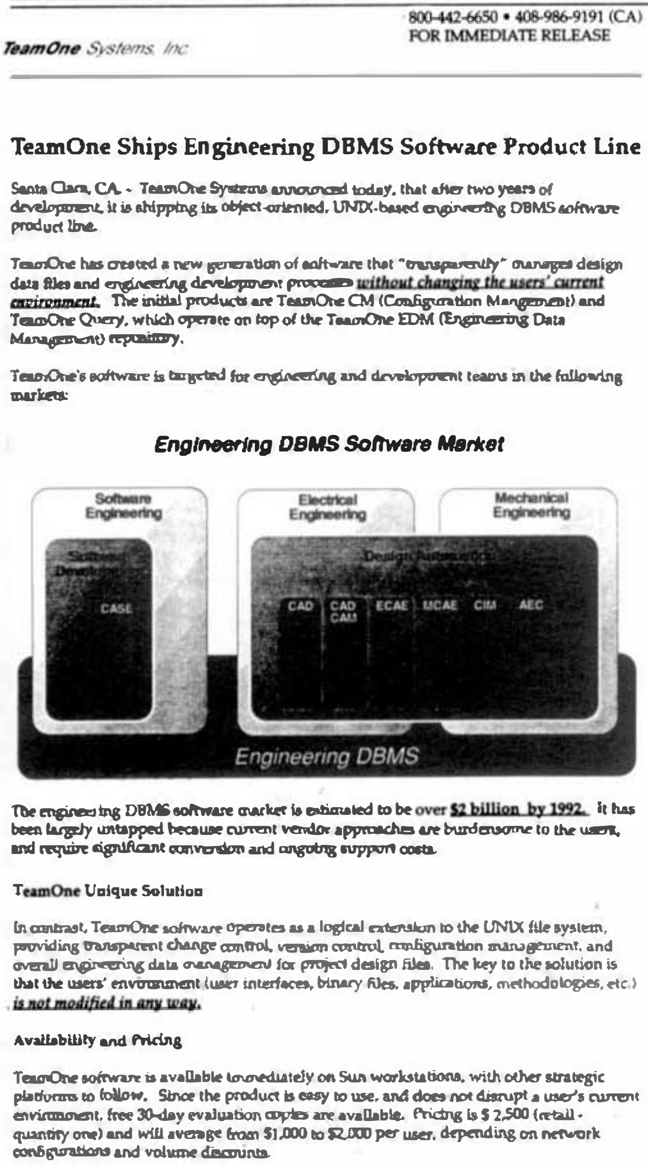
The most basic CPU-27 is priced as low as $4,490 but offers features associated with boards costing hundreds more. A fully configured 25 MHz CPU-27 is priced at $5,590. The family offers the following features:

- 68020 32-bit microprocessor, 12.5, 16.7 or 25 MHz operation; 2 EPROM sockets for up to 2 Mbytes of storage
- 68882 32-bit floating point coprocessor, 12.5, 16.7 or 25 MHz operation
- 1 Mbyte of high speed SRAM, 0 wait states at 12.5 MHz, 1 wait state at 16.7 MHz and 2 wait states at 25 MHz
- Real time clock with on-board battery backup
- 32 Kbyte single-socket SRAM with battery backup, can be used as a context store that survives power failure
- SCSI interface (via MB87031 host adapter chip)
- Optional Ethernet transceiver interface (front panel access) with 64 Kbyte dedicated buffer; based on the AMD 7990 LANCE chip set
- 3 serial I/O channels for RS232/RS422/RS485 operation available via 9-pin front panel connectors. First 2 ports can provide synchronous operation based on the 68562 dual universal serial communications controller chip; 3rd port employs the 68901 multi-function peripheral chip
- 2 Parallel Interface/Timers provide 38 parallel I/O lines on the P2 connector that can be programmed for uni- or bi-directional operation; timer functions include two 24-bit and eight 8-bit timers
- VME/PLUS architecture embodied in FGA-001, a 135-pin CMOS gate array. This array provides IEEE 1014 VMEbus interface and control functions including DSACK generation, bus error generation, system reset, bus clock and all on-board control logic.
- A32/24/16, D32/24/16/8 VMEbus master/slave interface
- Slot 1 Control functions (SYSCLOCK, arbiter, etc.)
- VMEPROM real-time PDOS kernel, monitor and debugger

### CPU-27 Software Includes VMEPROM

Enhancing the usefulness of the CPU-27 is VMEPROM, a free real-time operating system kernel that also includes a monitor and debugger. It is installed in EPROM, thus yielding operational capability as soon as the board is installed on an active backplane. VMEPROM is based on PDOS, a popular operating system from Eyring Research Institute.

The CPU-27 is also compatible with many third-party real-time systems and kernels, including UNIX-compatible products. Support for Ethernet's TCP/IP protocol is planned.



800-442-6650 • 408-986-9191 (CA)
FOR IMMEDIATE RELEASE

TeamOne Systems, Inc

## TeamOne Ships Engineering DBMS Software Product Line

Santa Clara, CA - TeamOne Systems announced today, that after two years of development, it is shipping its object-oriented, UNIX-based engineering DBMS software product line.

TeamOne has created a new generation of software that "transparently" manages design data files and engineering development process ***without changing the users' current environment.*** The initial products are TeamOne CM (Configuration Mangement) and TeamOne Query, which operate on top of the TeamOne EDM (Engineering Data Management) repository.

TeamOne's software is targeted for engineering and development teams in the following markets:

***Engineering DBMS Software Market***

The engineering DBMS software market is estimated to be over **$2 billion by 1992.** It has been largely untapped because current vendor approaches are burdensome to the users, and require significant conversion and ongoing support costs.

### TeamOne Unique Solution

In contrast, TeamOne software operates as a logical extension to the UNIX file system, providing transparent change control, version control, configuration management, and overall engineering data management for project design files. The key to the solution is that the users' environment (user interfaces, binary files, applications, methodologies, etc.) ***is not modified in any way.***

### Availability and Pricing

TeamOne software is available immediately on Sun workstations, with other strategic platforms to follow. Since the product is easy to use, and does not disrupt a user's current environment, free 30-day evaluation copies are available. Pricing is $ 2,500 (retail - quantity one) and will average from $1,000 to $2,000 per user, depending on network configurations and volume discounts.

## 68000 C CROSS-COMPILER

The 68000 C cross-compiler is a reasonably complete, optimizing, one-pass implementation of the version of C as described in the original Kernighan and Ritchie text, and as later extended.

This C compiler produces 68000 assembler code, which is then processed by a 68000 assembler conforming to the Motorola 68000 assembler language to produce 68000 machine language code for execution on a 68000 computer. Such an assembler is provided with this C compiler, although similar, system-specific assemblers may be substituted.

Since both the C compiler and assembler are themselves written in the C language, they are both capable of being ported to a number of systems. The only direct system dependence is that the provided C library was written for SK*DOS; however, the library is necessary only for running the program produced, not for running the C compiler and the assembler. This C library may be rewritten, replaced, or ignored, such as if code is being generated for stand-alone single-board applications.

The original Kernighan and Ritchie C text, and similar publications, such as the second edition of this text, comprise the best available references for the C language as embodied in this compiler. See B. W. Kernighan and D. M. Ritchie, "The C Progtamming Language", Prentice-Hall, 1978 and "The C Programming Language - Second Edition", Prentice-Hall, 1988.

The use of the compiler and/or the object code produced by it requires the following versions of operating systems:
MS-DOS 3.1 or greater
SK-DOS 2.5 or greater
UNIX BSD 4.2 or greater
UNIX System 5.2 or greater

The cross-compiler with printed manual is available for a retail price of $100. The C sources are available only by special arrangement. The program may be ordered from the following address:

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone Number 404-483-4570/1717

**From the February Issue page 45:**

### Writing Position Independent and Reentrant Code for the MC68000 Family by Truman T. Van Sickle

Our apologies for the omission of the last three paragraphs to the article.

...

This fixed code sequence can be moved anywhere in memory and it will execute correctly. Note that the table entries will be calculated at link time if the various procedures are in different modules. Also, it is not even necessary for the table and the executing code sequence to be in the same module so long as all of the necessary labels are properly defined with XREF and XDEF statements.

The above discussion outlined transfer of program control within a position indepetident program when the range to the destination code exceeds the plus or minus 32000 byte limit of the MC68000 family. A similar approach can be used to access memory values. These approached should be used sparingly. The code sequence above replaces a simple BSR operation. Only two of the eight instructions are required to pick a specific entry from the table. If the table contained only one entry, which corresponds to a simple BSR, five instructions would be needed.

Conclusion

The architecture of the MC68000 family of parts permits both reentrant and position independent code. These two important code features are easily obtained and there is minimal cost in terms of extra coding, memory usage, or program speed to obtain either feature. The only time that it becomes expensive memory or speed wise is then very long reaches to the destination in a position independent operation are needed. The range that the MC68000 can reach with no loss of effectiveness is plus or minus 32000 bytes.

Continued From Page 52 Dec.88

## DO - A FLEX-09 Batch File Processor

By: Dave Howland

```
      *
      *
      * SETEXT extension codes
      *
      X_BIN   EQU     0
      X_TXT   EQU     1
      X_CMD   EQU     2
      X_BAS   EQU     3
      X_SYS   EQU     4
      X_BAK   EQU     5
      X_SCR   EQU     6
      X_DAT   EQU     7
      X_BAC   EQU     8
      X_DIR   EQU     9
      X_PRT   EQU     10
      X_OUT   EQU     11
      *
      *
      * Error numbers
      *
      E_ILLG  EQU     1                illegal FMS function code
      E_OPEN  EQU     2                requested file in use on open
      E_EXST  EQU     3                file already exists on rename or
open for write
      E_FNF   EQU     4                file not found
      E_DFUL  EQU     7                disk full
      E_EOF   EQU     8                end of file encountered
      E_READ  EQU     9                disk file read error
      E_WRIT  EQU     10               disk file write error
      E_WPRT  EQU     11               file or disk is write protected
      E_DPRT  EQU     12               file is delete protected
      E_FCB   EQU     13               illegal file control block
      E_DNR   EQU     16               disk drive not ready
      E_STS   EQU     18               file error (read/write on wrong/
closed file)
      E_IDX   EQU     19               random byte number > 255
      E_FSPC  EQU     21               illegal filename specification
      E_OVFL  EQU     23               sector map overflow creating random
file
      E_NREC  EQU     24               non-existant record number specified
      E_EREC  EQU     25               record number match error, file
damaged
      E_SNTX  EQU     26               command syntax error
      E_PRNT  EQU     27               command not allowed while printing
      E_HRDW  EQU     28               hardware configuration error
      *
      *
      * Terminal drivers jump table addresses
      *
      T_INNE  EQU     $D3E5            input character, no echo
      T_IRQH  EQU     $D3E7            IRQ interrupt handler
      T_SWIV  EQU     $D3E9            SWI3 interrupt vector
      T_IRQV  EQU     $D3EB            IRQ interrupt vector
      T_TOFF  EQU     $D3ED            timer off
```

```
T_TON   EQU     $D3EF           timer on
T_TINT  EQU     $D3F1           timer initialise
T_MON   EQU     $D3F3           monitor entry
T_INIT  EQU     $D3F5           terminal initialisation
T_STAT  EQU     $D3F7           input status check
T_OUTC  EQU     $D3F9           output character
T_INCH  EQU     $D3FB           input character, with echo
*
*
* Disk driver entry points
*
D_READ  EQU     $DE00           read sector
D_WRIT  EQU     $DE03           write sector
D_VRFY  EQU     $DE06           verify sector
D_RSTR  EQU     $DE09           restore to track zero
D_DRIV  EQU     $DE0C           select drive
D_CHK   EQU     $DE0F           check drive ready
D_QCHK  EQU     $DE12           quick check drive ready
D_INIT  EQU     $DE15           initialise drivers
D_WARM  EQU     $DE18           warm start drivers
D_SEEK  EQU     $DE1B           seek track
*


*****************************************************************************
*
* DO.TXT - FLEX COMMAND FILE PROCESSOR
*
*****************************************************************************
*
* Command syntax :
*
*   DO <command file> {<parameter 1> <parameter 2> ... <parameter 9>}
*
*
* Command file syntax :
*
*   ECHO [ON | OFF]                           turn on/off echoing of command
lines
*   EXIT                                      exit from command file
*   GOTO <label>                              continue command file from
<label>
*   IF {NOT} <cond> GOTO <label>              goto <label> if condition met/not
met
*   IF {NOT} <cond> THEN..ELSE..ENDIF         compound IF structure
*   NOTE <message>                            output message
*   ON ERROR [CONTINUE | GOTO <label>]        clear/set error trap
*   PAUSE <message>                           output message and wait for key
input
*   REM <comment>                             comment
*   SHIFT                                     shift parameters 1 to 9 left
*   !<comment>                                comment
*   @<label>                            label
*
* IF conditions (<cond>) may be one of the following :
*
*   EXIST <file>                        condition true if <file> exists
*   ERROR                               condition true if an error has
occurred
*   <str> = <str>                       condition true if strings are
equal
*
* Substitution parameters are identified as follows :
*
*   %0          filename of command file
*   %1 - %9     command line parameters 1 to 9
*   %c          read one character from the terminal
*   %l          read a line from the terminal
*   %%          the character %
*
**************************************************************************
*
        OPT     PAG
        TTL     FLEX COMMAND FILE PROCESSOR
*
* INCLUDE FLEX LIBRARY HEADER FILE (NOT LISTED)
*
        OPT     NOL
        LIB     FLEX.H
        OPT     LIS
*
*
* CONSTANTS
*
NPARAM  EQU     10              MAXIMUM NUMBER OF PARAMETERS
BUFLEN  EQU     128             BUFFER AREA FOR PARAMETERS
BRKCHR  EQU     $03             BREAK CHARACTER
CR      EQU     $0D
LF      EQU     $0A
SPACE   EQU     $20
*
PARCON  EQU     1               PARAMETER VALUES
PARELS  EQU     2
PARNOP  EQU     3
PARERR  EQU     4
PARMST  EQU     5
PARGTO  EQU     6
PARIF   EQU     7
PARNOT  EQU     8
PAROFF  EQU     9
PARON   EQU     10
PARTHN  EQU     11
*
**************************************************************************
**************************************************************************
```

```
*
* TRANSIENT PART OF 'DO'
*
**************************************************************************
*
        SETDP
        ORG     $C100
*
DO      BRA     START
VN      FCB     1
        FCC     'Copyright 1987 DE Howland.'
        FCC     'May be copied for personal use only.'
*
* MESSAGES FOR TRANSIENT PART OF 'DO'
*
PARERM  FCC     'Parameters too long'
        FCB     4
PTRERM  FCC     'Too many parameters'
        FCB     4
*
DEFEXT  FCC     'BAT'           DEFAULT COMMAND FILE EXTENSION
*
*
START   STS     SAVESP
        TST     CMDFLG          IF NESTED 'DO' THEN SKIP RESIDENT INIT
        BNE     NESTED
        LDD     MEMEND          SAVE MEMEND
        STD     SVMEND
        SUBD    #ENDDO-RESIDO+1 SETUP NEW MEMEND ON PAGE BOUNDARY
        CLRB
        SUBD    #1
        STD     MEMEND
        STD     NEWMND
        LDA     TTYPS           SAVE TTY PAUSE
        STA     SVPAUS
        CLR     TTYPS           DISABLE PAUSE
        LDA     SPECIO          SAVE SPECIAL IO FLAG
        STA     SVSPIO
        LDA     #1              DISABLE ESCAPE PROCESSING
        STA     SPECIO
*
* INITIALISE VARIABLES
*
NESTED  CLR     INPARM          NOT EXPANDING PARAMETER
*
* READ COMMAND FILE NAME FROM COMMAND LINE
*
        LDX     #DOFCB          GET FCB POINTER
        JSR     GETFIL
        BCC     SPECOK
DSKER1  LBRA    DSKERR          REPORT DISK ERROR
*
SPECOK  TST     F_EXT,X         IF NO EXTENSION WAS SPECIFIED
```

```
        BNE     EXTOK
        IDD     DEFEXT          THEN SETUP DEFAULT EXTENSION '.BAT'
        STD     F_EXT,X
        LDA     DEFEXT+2
        STA     F_EXT+2,X
*
* OPEN COMMAND FILE FOR READ AND CLOSE IT AGAIN TO KEEP IT OUT OF THE FCB
CHAIN
*
EXTOK   IDA     #M_OPNR         OPEN FCB FOR READ
        STA     F_FUNC,X
        JSR     FMS
        BNE     DSKER1
        LDA     #M_CLOS         CLOSE IT AGAIN
        STA     F_FUNC,X
        JSR     FMS
        BNE     DSKER1
        LDA     #M_OPNR         SET ACTIVITIY STATUS = OPEN FOR READ
        STA     F_ACT,X
        LDA     #M_READ         SETUP TO READ BYTE
        STA     F_FUNC,X
*
* READ PARAMETERS FROM COMMAND LINE
*
        BSR     RDCMDL
*
* MOVE RESIDENT PART OF 'DO' TO TOP OF MEMORY
*
        LDX     #RESIDO         SOURCE POINTER
        LDY     MEMEND          DESTINATION POINTER
        LEAY    1,Y
MOVEDO  LOA     ,X+
        STA     ,Y+
        CMPX    #ENDDO
        BNE     MOVEDO
*
* IF NESTED 'DO' JUMP TO WARMS EISE JUMP TO 'DO' COMMAND LOOP
*
        TST     CMDFIG
        BEQ     GOTODO
        JMP     WARMS
*
GOTODO  LDX     MEMEND          JUMP TO RESIDENT PART OF 'DO'
        LEAX    1,X
        JMP     0,X
*
************************************************************************
*
* COMMAND LINE AND PARAMETER PROCESSING ROUTINES
*
************************************************************************
*
* READ COMMAND LINE INTO BUFFER AND PROCESS PARAMETERS
*
RDCMDL  BSR     INITCB          INITIALISE COMMAND BUFFER
        CLRB                    CLEAR LAST QUOTE FLAG
RDCM1   JSR     NXTCH           GET NEXT CHAR FROM COMMAND LINE
        CMPA    #CR             IF CR THEN EXIT
        BPQ     RDCMEX
        CMPA    #''             IF CHAR IS EITHER TYPE OF QUOTE
        BEQ     RDCM1B
        CMPA    #'"
        BNE     RDCM1A
RDCM1B  TFR     A,B             THEN SET IAST QUOTE
        JSR     NXTCH           GET NEXT CHAR
        CMPA    #CR             IF CR THEN EXIT
        BEQ     RDCMEX
RDCM1A  BSR     PUTPTR          ELSE HAVE START OF NEW PARAMETER
RDCM2   TSTB                    IF IAST QUOTE FIAG = 0
        BNE     RDCM2A
        CMPA    #SPACE          THEN IF CHAR = SPACE, HAVE SEPARATOR
        BEQ     RDCM4
RDCM2A  PSHS    B               IF CHAR = IAST QUOTE
        CMPA    ,S+
        BNE     RDCM3
        CIRB                    THEN CLEAR LAST QUOTE FLAG
        LDA     [LINPTR]        IF NEXT CHAR = SPACE
        CMPA    #SPACE
        BNE     RDCM4
        JSR     NXTCH           THEN GET IT
        BRA     RDCM4
*
RDCM3   BSR     PUTPAR          PUT CHAR INTO PARAMETER BUFFER
        JSR     NXTCH           GET NEXT CHAR FROM COMMAND LINE
        CMPA    #CR             IF CR THEN EXIT
        BEQ     RDCMEX
        BRA     ROCM2
*
RDCM4   BSR     ENDPAR          END OF PARAMETER; PUT ZERO
        BRA     ROCM1
*
RDCMEX  BSR     ENDPAR
        RTS
*
*
* PUT ZERO OR CHAR TO PARAMETER BUFFER
*
ENDPAR  CLRA
PUTPAR  CMPX    BUFFER+BUFIEN-1
        BEQ     PRMERR
        STA     ,X+
        RTS
*
PRMERR  LDX     #PARERM         OUTPUT PARAMS TOO LONG MESSAGE AND EXIT
        LBRA    PERROR
```

```
*
*
* SETUP PARAMETER POINTER
*
PUTPTR  PSHS    X,D
        LDY     CURPTR
        CMPY    #PARPTR+2*NPARAM
        BEQ     PTRERR
        LDD     MEMEND
        SUBD    #RESIDO-1
        LEAX    D,X
        STX     ,Y++
        STY     CURPTR
        PULS    X,D
        RTS
*
PTRERR  LDX     #PTRERM         OUTPUT TOO MANY PARAMS MESSAGE AND EXIT
        LBRA    PERROR
*
*
* INITIALISE COMMAND BUFFER AND POINTERS
* SET PARAMETER 0 -COMMAND FILE NAME
*
INITCB  LDX     #BUFFER+BUFLEN-1
        CLR     D,X             SETUP NULL PARAM AT END OF BUFFER
        LDY     #PARPTR+2
        LDB     #NPARAM
INITL1  STX     ,Y++            SET PARAM POINTERS 1 TO 9 -> NULL PARAM
        DECB
        BNE     INITL1
        LDX     #PARPTR+2       SET CURRENT PARAM POINTER -> PARAM 1
POINTER
        STX     CURPTR
        LDX     #BUFFER         SET PARAM POINTER 0 -> BUFFER
        LDD     MEMEND
        SUBD    #RESIDO-1
        LEAX    D,X
        STX     PARPTR
        LDX     #BUFFER
        LDY     #DOFCB+F_NAME   MOVE COMMAND FILE NAME INTO PARAM 0 IN
BUFFER
        LDB     #8
INITL2  LDA     ,Y+
        BEQ     INIT3
        STA     ,X+
        DECB
        BNE     INITL2
INIT3   CLR     ,X+             TERMINATE PARAM 0 WITH ZERO
        RTS
*
**************************************************************************
**************************************************************************
*
* RESIDENT PART OF 'DO'
*
**************************************************************************
*
        ORG     $100
*
RESIDO  LBRA    RESID
*
* VARIABLES
*
SAVESP  RMB     2               SAVED STACK POINTER
SVMEND  RMB     2               SAVED MEMEND
SVPAUS  RMB     1               SAVED TTY PAUSE FLAG
SVSPIO  RMB     1               SAVED SPECIAL IO FLAG
NEWMND  RMB     2               NEW MEMEND FOR EXIT CHECK
DOFCB   RMB     F_LEN           FCB FOR READING COMMAND FILE
BUFFER  RMB     BUFLEN          PARAMETER BUFFER
PARPTR  RMB     2*NPARAM+2      PARAMETER POINTERS
CURPTR  RMB     2               CURRENT PARAMETER POINTER
ERRLAB  RMB     9               ERROR LABEL (IF ERRFLG <> 0)
LABEL   RMB     9               TEMPORARY LABEL BUFFER
LABPTR  RMB     2               LABEL POINTER
TRMADR  RMB     2               ADDRESS OF LINE TERMINATOR
SAVECH  RMB     1               SAVED ECHO STATUS
*
INPARM  FCB     0               PARAMETER EXPANSION FLAG (0 = NOT IN
PARAM)
INLINE  FCB     0               TERMINAL LINE FLAG (0 = NOT TERMINAL)
INTEXT  FCB     0               TERMINAL TEXT FLAG (0 = NOT TERMINAL)
STLINE  FCB     $FF             START OF LINE FLAG (NEXT CHAR = START OF
LINE)
IFLEVL  FCB     0               CURRENT 'IF' NESTING LEVEL
IFLEVX  FCB     0               BYPASS IF NESTING LEVEL
ERRFLG  FCB     0               ERROR HANDLING FLAG (DEFAULT = CONTINUE)
LSTERR  FCB     0               LAST DOS ERROR
ECHOFL  FCB     $FF             COMMAND FILE ECHO FLAG (DEFAULT = ECHO)
GOTOFL  FCB     0               GOTO FLAG (0 = NOT LOOKING FOR LABEL)
NOTFLG  FCB     0               'NOT' PARAM FLAG (0 = 'IF', $FF = 'IF
NOT')
*
* COMMAND TABLE
*
CTABLE  FCC     'ECHO'
        FCB     0
        FDB     DOECHO-CTABLE
        FCC     'ELSE'
        FCB     0
        FDB     DOELSE-CTABLE
        FCC     'ENDIF'
        FCB     0
        FDB     DOENDF-CTABLE
        FCC     'EXIT'
        FCB     0
        FDB     DOEXIT-CTABLE
```

To Be Continued

## Classifieds

As Submitted - No Guarantees

**Surplus Unused Motorola VME Modules & Electronic Solutions Enclosures for Sale at Discount**

| | | |
|---|---|---|
| MVME133 | CPU Module-68020, 1MB DRAM, 68881 FPP,<br>3 serial Ports, EPROM Sockets, VMEbus Interface | $675 |
| MVME225-1 | 1MB DRAM Module, A32/D32 VMEbus Interface | $380 |
| MVME320A | Winchester / Fllopy Controller | $490 |
| MVME332 | 8 Channel intelligent Serial Communications Module | $675 |
| Series 7 | Electronic Solutions 7 Slot Desktop Enclosure, P1/P2<br>Backplane, 325W PS, Space for Winchester/Floppy/Tape | $695 |

**Respond to:** John Gannon, RPG, P.O. Box C12399, Ste 162, Scottsdale, Arizona 85267 Phone (602) 951-3373

***

S+ Memory Cards, CPU Cards, Hard Disks w/Controller Cards, I/O Cards, Cabinets, Power Supplies.
S/09 CPU Cards, Memory, I/O Cards, Controller Cards, Cabinets, Power Supplies
3-Dual 8" drive enclosure with power supply. New in box. $125 each.
5-Siemens 8" Disk Drive, $100 each.

**Tom (615) 842-4600 M-F 9AM to 5PM EST**

***

**Motorola VME-10** with hi resolution monochrome monitor, hard disk, serial and parallel cards, Pascal, assembler, linker and documentation. Almost new $4500.
Two SSB-6809 systems with hard disks, miscellaneous software. Make Offer
**Cadwell Laboratories**, 909 N. Kellogg Street, Kennewick, WA 99336, **(509) 735-6431**

**FLEX™/SK-DOS™/MS-DOS™**

**Transfer Utilities**

**For 68XXX and CoCo* OS-9™ Systems**

**Now READ - WRITE - DIR - DUMP - EXPLORE**
**FLEX, SK-DOS & MS-DOS Disk**

These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks.

*CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.

CoCo Version: $69.95 68XXX Version $99.95

**S.E. Media** ←→ **PO Box 849, Hixson,TN 37343**

**615 842-6809** **MC/Visa**

## SK*DOS®/68K

Read the fine print to see what's in SK* S/68K:

□ Full DOS documentation plus on-line help □ Multiple directories □ User-installable device drivers □ Install up to 8 different I/O devices □ Keyboard type-ahead □ Print-screen □ Virtual (RAM) disk □ Disk cache □ Up to 10 drives □ 5¼" or 3½" floppy drives □ Hard drives to 64 megabytes each □ I/O redirection to drives or I/O □ Time/date stamping of files □ File or disk write protect (even hard disk) □ Batch files □ Support for 68000, 68010, 68020 □ Monochrome or color video board support □ Read and write MS-DOS disk files □ 6809 Emulator □ Powerful utilities such as copy-by-date, undelete, show differences between files, prompted delete, text file browse, and more - all included □ Simple Basic included □ Fast assembler included □ Line editor included □ User support via newsletter and BBS □ Available software: C compiler, full Basic, screen editors, disassemblers, cross assemblers, spelling checker, text formatter, music editor, hard disk manager, ROM-based debugger, modem communications programs, etc. More compilers coming. □ (Some features may not be implemented in all hardware manufacturers' implementations.)

Individual copies of SK*DOS/68K are $140; less in quantity or when bundled with hardware. Send for our 6809 / 68K hardware and software catalog. Also available as part of our hardware/software educational course.

STAR-K Software Systems Corp.
P. O. Box 209J
Mt. Kisco, NY 10549
(914) 241-0287
BBS (914) 241-3307 ◊ Fax (914) 241-8607

## 68030 Modula–2

**OS-9 or PC Host, 880x0/6888x Target**

- Very fast compilation - 10500 lpm (20MHz 68020)
- Very fast execution - highly optimised compiler
- Object code half the size of other 68k compilers
- Library support for all OS-9 system calls
- All 680X0 series target processors supported
- Source code tools and library source provided
- Run-time debugger and ROM tools available
- 500 pages of comprehensive documentation

*Modula-2 compiler, OS-9 or PC host:* **$1790.00**
*ROM tools:* **$720.00** *Run-time debugger:* **$720.00**

## Generic Lint

**OS-9 Multi-file C source code checker**

Detects errors, quirks, idiosyncracies and likely mistakes in multiple source and header files. Cuts debugging time by cutting out bugs ! *Generic Lint* **$398.00**

Call for information on OS-9 consultancy services.
Trademarks: 'OS-9' Microware, 'Generic Lint' Gimpel

Windsor Systems
2407 Lime Kiln Lane, Louisville, KY 40222 U.S.A
Telephone: (502) 426 9660 Fax: (502) 426 3944

## SOFTWARE

**68000 C CROSS-COMPILER**
*$100 - SKDOS,MSDOS,UNIX,XENIX (OBJECT ONLY)*

Accepts K&R C language, generates 68000 assembler code
includes 68010 cross-assembler, libraries provided for SKDOS, but may be modified

**CROSS-ASSEMBLERS WITH MACRO CAPABILITIES**
*EACH $50-FLEX,OS9,UNIFLEX,MSDOS-UNIX,SKDOS,XENIX 3/$100 ALL/$200*

Specify: 180x, 6502, 6801/11, 6804, 6805, 6809, Z8, Z80, 8048, 8051, 8085, 68010, 32000
Modular cross-assemblers in C, with load/unload utilities. Sources for additional $50 each, $100 for 3, $300 for all

**CMODEM TELECOMMUNICATIONS PROGRAM**
*$100-MSDOS,SKDOS,UNIX,FLEX,OS9,XENIX,UNIFLEX OBJECT-ONLY: EACH $50*

Menu-driven with terminal mode, file transfer, MODEM7, XON-XOFF, etc

**SUPER SLEUTH DISASSEMBLERS**
*EACH $99-FLEX $101-OS9 $100-UNIFLEX OBJECT-ONLY: EACH $50 FLEX,OS9,COCO*

Interactively generate source on disk with labels, includes xref, binary editing
Specify 6800,1,2,3,5,8,9/6502 version or Z80/8080,5 version
COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not Z80/8080,5) only
68010 version $100-FLEX,OS9,UNIFLEX,MSDOS,UNIX,SKDOS,XENIX

**DEBUGGING SIMULATORS FOR POPULAR 8-BIT CPUS**
*EACH $75-FLEX $100-OS9 $80-UNIFLEX OBJECT-ONLY: EACH $50-COCO FLEX, COCO OS9*

Interactively simulate processors, includes disassembly formatting, binary editing
Specify for 6800/1, (14)6805, 6502, 6809 OS9 only, Z80 FLEX only

**ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809**
*6502 to 6809 $75-FLEX $85-OS9 $80-UNIFLEX*
*6800/1 to 6809 & 6809 to pos.ind. $50-FLEX $75-OS9 Only $60-UNIFLEX*

**FULL-SCREEN XBASIC PROGRAMS with cursor control**
**AVAILABLE FOR FLEX, UNIFLEX, AND MSDOS**

| | | |
|---|---|---|
| Display Generator / Documenter | $50 w/source, | $25 without |
| Mailing List System | $100 w/source, | $50 without |
| Inventory with MRP | $100 w/source, | $50 without |
| Tabula Rasa Spreadsheet | $100 w/source, | $50 without |

**DISK AND XBASIC UTILITY PROGRAM LIBRARY**
*$50-FLEX $30-UNIFLEX/MSDOS*

Edit disk sectors, sort directory, maintain master catalog, do disk sorts, resequence some or all BASIC program, xref BASIC program, etc. non-FLEX versions include sort and resequencer only

## PROFESSIONAL SERVICES
FOR THE COMPUTING COMMUNITY

*CUSTOMIZED PROGRAMMING*

We will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications.

*CONTRACT PROGRAMMING*

We will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 680x0, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.

CONSULTING

We offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses

**Computer Systems Consultants Inc.**
1454 Lakeview Lane
Conyers, Georgia 30207
(404) 483-4570 • (404) 483-1717

Contact us about catalog, dealer, discounts, and services. Most programs in source: give computer, OS, disk size. 25% off multiple purchases of same program on one order. VISA and MASTER CARD accepted. Add GA sales tax (if in GA) and 5% shipping. (UNI)FLEX tm Technical Systems Consultants; OS/9 Microware COCO Tandy;MSDOS Microsoft; SKDOS Stark Software

Clearbrook Software Group (604)853-9118

CSG IMS

CSG IMS is *THE* full featured relational database manager for OS9/OSK. The comprehensive structured application language and B+Tree index structures make CSG IMS the ideal tool for file-intensive applications.

| | |
|---|---|
| CSG IMS for CoCo2/3 OS9 L1/2 (single user) | $169.95 |
| CSG IMS for OS9 L2 or 68000(multi user) | $495.00 |
| CSG IMS demo with manual | $30 |

**MSF - MSDos File Manager for CoCo 3/OS9 Level 2**

allows you to use MSDos disks directly under OS9.

Requires CoCo 3, OS9 L2, SDISK3 driver $45.00

**SERINA - System Mode Debugger for OS9 L2**

allows you to trace execution of any system module, set break points, assemble and disassemble code and examine and change memory.

Requires CoCo3 or Gimix II, OS9 L2 & 80 col. terminal $139.00

**ERINA - Symbolic User Mode Debugger for OS9**

lets you find bugs by displaying the machine state and Instuctions being executed. Set break points, change memory, assemble and disassemble code.

Requires 80 column display, OS9 L1/2 $69.00

Shipping: N. America - $5, Overseas - $10

Clearbrook Software Group P.O. Box 8000-499, Sumas, WA 98295

OS9 is a trademark of Microware Systems Corp., MSDos is a trademark of Microsoft Corp.

SPECIAL

ATARI™

&

OS-9™

NOW!
If you have either the
Atari 520 or 1040 -
you can take
advantage of the
*"bargain of a lifetime"*
OS-9 68K and BASIC
all for the low, low price of:

$150.00

Call or Write
S.E. Media
5900 Cassandra Smith Rd.
Hixson, TN 37343
615 842-4601

# *ATARI & AMIGA CALL*

As most of you know, we are very sensitive to your wishes, as concerns the contents of these pages. One of the things that many of you have repeatedly written or called about is coverage forthe **Atari & Amiga™** series of 68000 computers.

Actually we haven't been too keen on those systems due to a lack of serious software. They were mainly expensive "game-toy" systems. However, recently we are seeing more and more honest-to-goodness serious software for the Atari & Amiga machines. That makes a difference. I feel that we are ready to start some serious looking into a section for the Atari & Amiga computers. Especially so since OS-9 is now running on the Atari (review copy on the way for evaluation and report to you) and rumored for the Amiga. Many of you are doing all kinds of interesting things on these systems. By sharing we all benefit.

**This I must stress - *Input from you on the Atari & Amiga.* As most of you are aware, we are a "contributor supported" magazine. That means that YOU have to do your part. Which is the way it has been for over 10 years. We need articles, technical, reviews of hardware and software, programming (all languages) and the many other facets of support that we have pursued for these many years. Also I will need several to volunteer to do regular columns on the Atari & Amiga systems. Without constant input we can't make it fly! So, if you do your part, we certainly will do ours. How about it, drop me a line or give me a phone call and I will get additional information right back to you. Weneed your input andsupportif this is tosucceed!**

**DMW**

# THE 6800-6809 BOOKS

## ..HEAR YE.....HEAR

# OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles **OS9 USER NOTES**

**Information for the BEGINNER to the PRO, Regular or CoCo OS9**

**Using OS9**
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS. Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

**Program interfacing to OS9**
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

**Programming Languages**
Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

**Disks Include**
**No typing all the Source Listinga in.** Source Code and, where applicable, aasembled or compiled Operating Programs. The Source and the Discussions in the Columns can be used "aa ia", or as a "Starting Point" for developing your OWN more powerful Programs. Programs sometimea uae multiple Languagea such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

## BOOK $9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
**Deluxe Binder** - - - - - - - - - - - $5.50

### All Source Listings on Disk

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

# FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderaon's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the moat popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

| | |
|---|---|
| LOGO.C1 | File load program to offset memory — ASM PIC |
| MEMOVE.C1 | Memory move program — ASM PIC |
| DUMP.C1 | Printer dump program — uses LOGO — ASM PIC |
| SUBTEST.C1 | Simulation of 6800 code to 6809, show differences — ASM |
| TERMEM.C2 | Modem input to disk (or other port input to disk) — ASM |
| M.C2 | Output a file to modem (or another port) — ASM |
| PRINT.C3 | Parallel (enhanced) printer driver — ASM |
| MODEM.C2 | TTL output to CRT and modem (or other port) — ASM |
| SCIPKG.C1 | Scientific math routines — PASCAL |
| U.C4 | Mini-monitor, disk resident, many useful functions — ASM |
| PRINT.C4 | Parallel printer driver, without PFLAG — ASM |
| SET.C5 | Set printer modes — ASM |
| SETBAS1.C5 | Set printer modes — A-BASIC |

NOTE: .CI,.C2, etc.=Chapter I, Chapter 2, etc.

**Over 30 TEXT files included is ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

Shipping & Handling $3.50 per Book, $2.50 per Disk set
Foreign Orders Add $4.50 Surface Mail
or $7.00 Air Mail
If paying by check - Please allow 4-6 weeks delivery
* All Currency in U.S. Dollars

## Continually Updated In 68 Micro Journal Monthly

**Computer Publishing Inc.**
**5900 Cassandra Smith Rd.**
**Hixson, TN 37343**

VISA

**(615) 842-4601**
**Telex 5106006630**

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware and Motorola
"68' Micro Journal is a trademark of Computer Publiahing Inc.

# !!! Subscribe Now !!!
# 68 MICRO JOURNAL

## OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Mastercard □ VISA □

Card #________________________ Exp. Date_________

For 1 Year ____ 2 Years ____ 3 Years ____

Enclosed: $____________

**Name** ________________________________

**Street** ________________________________

**City** __________________ **State** ____ **Zip** ________

**Country** ________________________________

**My Computer Is:** ________________

### Subscription Rates

**U.S.A.: 1 Year $24.50, 2 Years $42.50, 3 Years $64.50**

***Foreign Surface: Add $12.00 per Year to USA Price.**

***Foreign Airmail: Add $48.00 per Year to USA Price.**

***Canada & Mexico: Add $9.50 per Year to USA Price.**

***U.S. Currency Cash or Check Drawn on a USA Bank !**

**68 Micro Journal**
5900 Cassandra Smith Rd.
POB 849
Hixson, TN 37343

**Telephone 615 842-4600**
**Telex 510 600-6630**

## Reader Service Disks

| | |
|---|---|
| Disk- 1 | Filesort, Minicat, Minicopy,Minifms, **Lifetime, **Poetry, **Foodlist, **Diet. |
| Disk- 2 | Diskedit w/ inst.& fixes, Prime, *Prmod, **Snoopy, **Football, **Hexpawn,**Lifetime. |
| Disk- 3 | Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave. |
| Disk- 4 | Mailing Program, *Finddat, *Change, *Testdisk. |
| Disk- 5 | *DISKFIX I, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING. |
| Disk- 6 | **Purchase Order, Index (Disk file indx). |
| Disk- 7 | Linking Loader, Rload, Harkness. |
| Disk- 8 | Crtest, Lanpher (May 82). |
| Disk- 9 | Datecopy, Diskfix9 (Aug 82). |
| Disk-10 | Home Accounting (July 82). |
| Disk-11 | Dissembler (June 84). |
| Disk-12 | Modem68 (May 84). |
| Disk-13 | *Initmf68, Testmf68,*Cleanup,*Dskalign,Help,Date.Txt. |
| Disk-14 | *Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib. |
| Disk-15 | Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo). |
| Disk-16 | Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc. |
| Disk-17 | Match Utility, RATBAS, A Basic Preprocessor. |
| Disk-18 | Parse.Mod, Size.Cmd (Sept. 85 Armstrong),CMDCODE, CMD.Txt (Sept. 85 Spray). |
| Disk-19 | Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc. |
| Disk-20 | UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, LS.C, FDUMP.C. |
| Disk-21 | Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more. |
| Disk-22 | Read CPM & Non-FLEX Disks. Fraser May 1984. |
| Disk-23 | ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven, Language Recognition Utility, Anderson March1986. |
| Disk-24 | 68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current. |
| Disk-25 | KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk. |
| Disk-26 | Compacta UniBoard review, code & diagram, Burlison March '86. |
| Disk-27 | ROTABIT.TXT, SUMSTEST.TXT, CONDATA.TXT, BADMEN.TXT. |
| Disk-28 | CT-82 Emulator, bit mapped. |
| Disk-29 | **Star Trek |
| Disk-30 | Simple Winchester, Dec.'86 Green. |
| Disk-31 | *** Read/Write MS/PC-DOS (SK*DOS) |
| Disk-32 | Heir-UNIX Type upgrade - 68MJ 2/87 |
| Disk-33 | Build the GT-4 Terminal - 68MJ 11/87 Condon. |
| Disk-34 | FLEX 6809 Diagnostics, Disk Drive Test, ROM Test, RAM Test - 68MJ 4/88Koipi. |

**NOTE:**

This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC
*** Denotes 68000 - 6809 no indicator.

**8" disk $19.50**
**5" disk $16.95**

Shipping & Handling -U.S.A. Add: - $3.50
Overseas add: $4.50 Surface - $7.00 Airmail

**68 MICRO JOURNAL**
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4600 - Telex 510 600-6630

# K-BASIC™

The Only 6809 BASIC to Binary Compiler for OS-9
FLEX or SK*DOS
Even runs on the 68XXX SK*DOS Systems*

*Hundreds Sold at Suggested Retail:*
~~$199.00~~

- 6809 - OS-9™ users can now transfer their FLEX™ Extended BASIC (XBASIC) source files to OS-9, compile with the OS-9 version and run them as any other OS-9 binary "CMD" program. Much faster than BASIC programs.

- 6809 - FLEX users can compiler their BASIC source files to a regular FLEX ".CMD" file. Much faster execution.

- 68XXX - SK*DOS™ users running on 68XXX systems (such as the Mustang-08/A) can continue to execute their 6809 FLEX BASIC and compiled programs while getting things ported over to the 68XXX. SK*DOS allows 6809 programs to run in emulation mode. This is the only system we know of that will run both 6809 & 68XXX binary files.

K-BASIC is a true compiler. Compiling BASIC 6809 programs to binary command type programs. The savings in RAM needed and the increased speed of binary execution makes this a must for the serious user. And the price is now RIGHT!

**Don't get caught up In the *"Learn a New Language"* syndrome - Write Your Program in BASIC, Debug it In BASIC and Then Compile It to a .CMD Binary File.**

**For a LIMITED time save over 65%... This sale will not be repeated after it's over! ***

**SALE SPECIAL:**
**$69.95**

# SPECIAL
## Thank-You-Sale

*Only From:*

**S.E. Media™**

CPI

5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

A Division of Computer Publishing Inc.
Over 1,200 Titles - 6800-6809-68000

* K-BASIC will run under 68XXX SK*DOS in emulation mode for the 6809.
Price subject to change without notice.

# PT-68000 SINGLE BOARD COMPUTER

The PT68K2 is Available in a Variety of Formats
From Basic Kits to Completely Assembled Systems

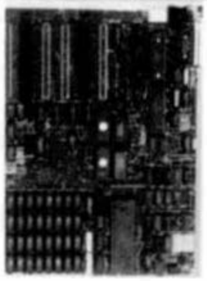
**BASIC KIT (8 MHZ)** - Board, 68000, HUMBUG MONITOR + BASIC in ROM, 4K STATIC RAM, 2 SERIAL PORTS, all Components $200

**PACKAGE DEAL** - Complete Kit with Board 68000 10 MHZ, SK*DOS, 512K RAM, and all Necessary Parts $575

**ASSEMBLED BOARD** (12 MHZ) Completely Tested, 1024K RAM, FLOPPY CONTROLLER, PIA, SK*DOS $899

**ASSEMBLED SYSTEM** - 10 MHZ BOARD, CABINET POWER SUPPLY, MONITOR + KEYBOARD, 80 TRACK FLOPPY DRIVE, CABLES $1299
For A 20 MEG DRIVE, CONTROLLER and CABLES Add $295

**PROFESSIONAL OS9** $500



### FEATURES

- MC68000 Processor, 8 MHZ Clock (optional 10,12.5 MHZ)
- 512K or 1024K of DRAM (no wait states)
- 4K of SPAM (6116)
- 32K,64K or 128K of EPROM
- Four RS-232 Serial Ports
- Floppy disk controller will control up to four 5 1/4", 40 or 80 track.
- Clock with on-board battery.
- 2 - 8 bit Parallel Ports
- Board can be mounted in an IBM type PC/XT cabinet and has a power connector to match the IBM type power supply.
- Expansion ports - 6 IBM PC/XT compatible I/O ports. The HUMBUG™ monitor supports monochrome and/or color adaptor cards and Western Digital winchester interface cards.

## PERIPHERAL TECHNOLOGY

1710 Cumberland Point Dr., Suite 8
Marietta, Georgia 30067
404/984-0742
VISA/MASTERCARD/CHECK/C.O.D.

Send For Catalogue
For Complete Information On All Products

™SK*DOS is a Trademark of STAR-K SOFTWARE SYSTEMS CORP.
™OS9 is a Trademark of Microware

# *DATA-COMP* SPECIAL

## Heavy Duty Power Supplies

**For A limited time our HEAVY DUTY SWITCHING POWER SUPPLY. These are BRAND NEW units. Note that these prices are less than 1/4 the normal price for these high quality units.**

Make: Boschert
Size: 10.5 x 5 x 2.5 inches
Including heavy mounting bracket and heatsink.

Rating: in 110/220 volts ac (strap change) Out: 130 watts

Output: +5v - 10 amps
+12v - 4.0 amps
+12v - 2.0 amps
-12v - 0.5 amps

Mating Connector: Terminal strip
Load Reaction: Automatic short circuit recovery
**SPECIAL: $59.95 each**
**2 or more $49.95 each**
Add: $7.50 each S/H

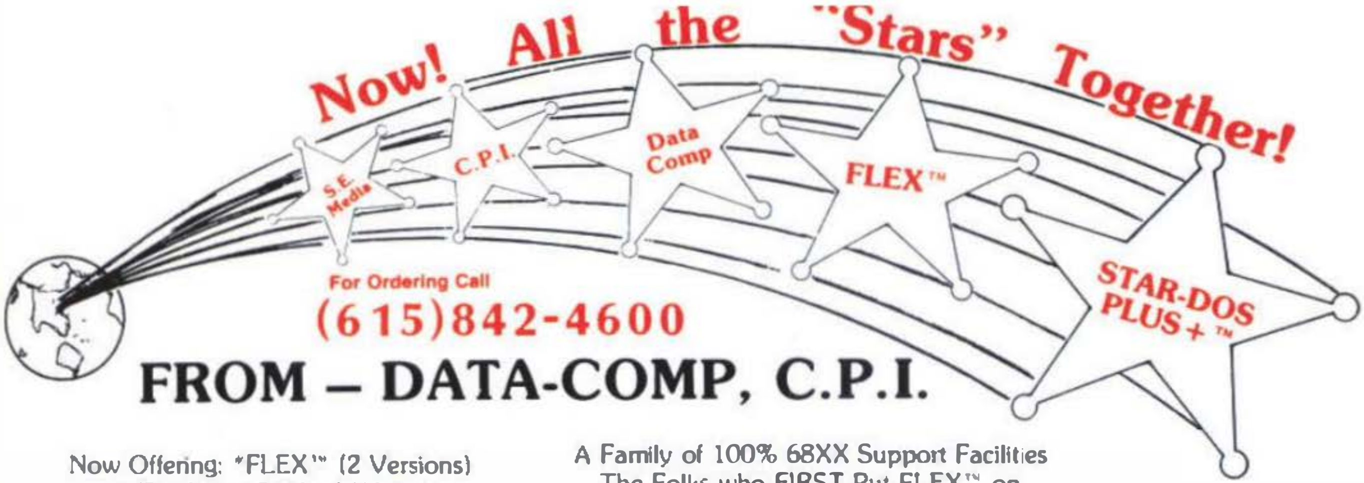
Make: Boschert
Size: 10.75 x 6.2 x 2.25 inches

Rating: 110/220 ac (strap change) Out: 81 watts

Outputs: +5v - 8.0 amps
+12v - 2.4 amps
+12v - 2.4 amps
+12v - 2.1 amps
-12v - 0.4 amps

Mating Connectors: Molex
Load Reaction: Automatic short circuit recovery
**SPECIAL: $49.95 each**
**2 or more $39.95 each**
Add: $7.50 S/H each

**5900 Cassandra Smith Rd., Hixson, Tn. 37343 Telephone 615 842-4600 Telex 510 600-6630**



# Now! All the "Stars" Together!

For Ordering Call

**(615)842-4600**

## FROM – DATA-COMP, C.P.I.

Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS + ™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

**FLEX-CoCo Sr.**
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. '250.$^{00}$ **Only '79.$^{95}$**

**STAR-DOS PLUS +**
- Functions Same as FLEX
- Reads - writes FLEX Disks '34.$^{50}$
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

**FLEX-CoCo Jr.**
without TSC
Editor & Assembler
'49.$^{95}$

**PLUS**

**ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE**

TSC Editor
Reg $50.00
**NOW $35.00**

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Assembler
Reg $50.00
**NOW $35.00**

### UPS Battery Backup

| | | |
|---|---|---|
| 330 | VA UPS 4 Outlets | 309.00 |
| 520 | VA UPS 4 Outlets | 599.00 |
| 800 | VA UPS 6 Outlets | 859.00 |
| 1200 | VA UPS 6 Outlets | 999.00 |

### Disks

5" Box of ten (10)

| | | |
|---|---|---|
| DS-DD | 5.99 | 40 Track |
| DS-DD | 14.95 | 80 Track |

3.5 DS-DD $15.95

### Printers

EPSON

| | | |
|---|---|---|
| LQ-500 | 24 Pin | 350.$^{00}$ |
| LX-800 | 9 Pin | 239.$^{00}$ |

### Gender Changer

| | |
|---|---|
| SERIAL | |
| Male to Male | 8.95 |
| Female to Female | 8.95 |
| Centronics | |
| Male to Male | 9.95 |
| Female to Female | 9.95 |

### Switch Boxes

| | |
|---|---|
| Serial/Parallel Converter | 69.00 |
| Parallel Serial Converter | 69.00 |
| Serial 2 Position Switch Box | 27.00 |
| Parallel 2 Position Switch Box | 27.00 |
| Serial 4 Position Switch Box | 35.00 |
| Parallel 4 Position Switch Box | 33.00 |
| Serial 4 Position Crossover Switch | 44.00 |
| Parallel 4 Position Crossover Switch | 49.00 |
| Serial 2 Position 9 Pin Switch Box | 44.00 |

**DATA-COMP**
**5900 CASSANDRA SMITH RD.**
**HIXSON, TN. 37343**

**SHIPPING**
**USA ADD 2%**
**FOREIGN ADD 5%**
**MIN $3.50**

VISA
master card

**(615) 842-4600**
**FOR ORDERING**
**TELEX 5106006630**

## *An Ace of a System in Spades! The New*

# MUSTANG-08/A™

**Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the $500.00 OS-9 C compiler! This offer won't last forever!**

## *NOT* 128K, *NOT* 512K
## *FULL* 768K No Wait RAM

The MUSTANG-08™ system took every hand from all other 68008 sysems we tested, running OS-9 68K!

The MUSTANG-08 includes OS9-68K™ and/or Peter Stark's SK*DOS™. SK*DOS is a single user, single tasking system that takes up where *FLEX™ left off. SK*DOS is actually a 68XXX FLEX type system *(Not a TSC product.)*

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It is a speed *whiz on disk I/O.* Fact is: the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that is just a small part of the story! See benchmarks.

SALE!

System includes OS-9 68K or SK*DOS - Your Choice

Specifications:

| | | |
|---|---|---|
| CPU | MC68008 | 12 Mhz |
| RAM | 768K | 256K Chips |
| | No Wait States | |
| PORTS | 4 - RS232 | MC68681 DUART |
| | 2 - 8 bit Parallel | MC6821 PIA |
| CLOCK | MK48T02 | Real Time Clock Bat. B/U |
| EPROM | 16K, 32K or 64K | Selectable |
| FLOPPY | WD1772 | 5 1/4 Drives |
| HARD DISK | Interface Port | WD1002 Board |

**Now more serial ports - faster CPU Battery B/U - and $850.00 OS-9 Professional with C compiler included!**

***$400.00** See Mustang-02 Ad - page 5 for trade-in details

Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68030

MUSTANG-08

LOOK

```
                                      Seconds   32 bit    Register
                                                Integer   Long
Other      68008   8   Mhz   OS-9   68K...18.0...9.0
MUSTANG-08  10   Mhz   OS-9   68K....9.8...6.3
                 Main()
                    {
                              /* int i; */
C Benchmark Loop              register long i;
                              for (i=0; i < 999999; ++i);
                    }
```

**Now even faster! with 12 Mhz CPU**

| C Compile times: OS-9 68K Hard Disk | |
|---|---|
| MUSTANG-08 8 Mhz CPU | 0 min - 32 sec |
| Other popular 68008 system | 1 min - 05 sec |
| MUSTANG-020 | 0 min - 21 sec |

♠ **25 Megabyte Hard Disk System**

# $2,398.90

**Complete with PROFESSIONAL OS-9 includes the $500.00 C compiler, PC style cabinet, heavy duty power supply, 5' DDDS 80 track floppy, 25 MegByte Hard Disk - Ready to Run**

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be easily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (system must have 128K in addition to its other requirements). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC          MUSTANG-08 is a trademark of CPI

## Data-Comp Division

A Decade of Quality Service"
Systems World-Wide

CPI
Computer Publishing, Inc.
5900 Cassandra Smith Road
Hixson, Tn 37343
Telephone 615 842-4601 - Telex 510 600-6630

* Those with SWTPC hi-density FLEX 5' - Call for special info.